Control de comportament d'un robot de servei per mitjà d'una arquitectura cognitiva

Autor:
Albert Pumarola Peris

Director:
Cecilio Angulo Bahón (ESAII)

Data de defensa: 30/06/2014
Titulació: Grau en Enginyeria Informàtica
Especialitat: Computació
Centre: Facultat d'Informàtica de Barcelona (FIB)
Universitat: Universitat Politècnica de Catalunya · UPC BarcelonaTech

Juny 2014

Agraïments

Voldria agrair a Ceciclio Angulo, director del treball, pel seu excel·lent suport i guia durant la realització d'aquest treball. També m'agradaria agrair l'empresa PAL Robotics i tots els membres de l'equip REEM@IRI, en especial a Ricardo Téllez per donar-me l'oportunitat d'involucrar-me en la robòtica humanoide de servei.

Resum

En el present Treball de Final de Grau es presenta un dels problemes més comuns en la robòtica de servei, resoldre tasques complexes. Aquest problema s'ha intentat resoldre proposant un nou mètode basat en arquitectures cognitives.

Així doncs, s'ha implementat una arquitectura cognitiva basada en *Soar*. Suposant un robot de servei dotat de múltiples habilitats com desplaçar-se, reconèixer objectes, etc. El robot, utilitzant aquestes habilitats, ha de completar una tasca complexa donada en llenguatge natural que requereix aquestes habilitats per a ser resolta. Gràcies a l'arquitectura implementada pot resoldre-la. *Soar* actua com a raonador que selecciona en cada moment l'acció a ser executada pel robot per completar la tasca.

Aquest sistema ha estat testejat en el robot de servei REEM en entorns domèstics i els resultats obtinguts mostren que l'arquitectura implementada permet resoldre tasques complexes com per exemple "bring me a drink". A més a més, també es demostra la gran canviabilitat de l'arquitectura i la seva capacitat de ser instal·lada en qualsevol robot independentment del seu hardware.

Resumen

En el presente Trabajo Final de Grado se presenta uno de los problemas más comunes en la robótica de servicio, resolver tareas complejas. Este problema se ha intentado resolver proponiendo un nuevo método basado en arquitecturas cognitivas.

Así pues, se ha implementado una arquitectura cognitiva basada en *Soar*. Suponiendo un robot de servicio dotado de múltiples habilidades como desplazarse, reconocer objetos, etc. El robot, utilizando estas habilidades, tiene que completar una tarea compleja dada en lenguaje natural que requiere de estas habilidades para ser resuelta. Gracias a la arquitectura implementada puede resolverla. *Soar* actúa como razonador que selecciona en cada momento la acción a ser ejecutada por el robot para completar la tarea.

Este sistema ha sido probado en el robot de servicio REEM en entornos domésticos y los resultados obtenidos muestran que la arquitectura implementada permite resolver tareas complejas como por ejemplo "bring me a drink". Además, también se demuestra la gran cambiabilidad de la arquitectura y su capacidad de ser instalada en cualquier robot independientemente de su hardware.

Abstract

In this Final Degree Project it is presented one of the most common problems related to service robotics, solving complex tasks. To solve this problem a new method based on cognitive architectures is proposed.

Therefore, a cognitive architecture based on *Soar* has been implemented. Assuming a a humanoid service robot is equipped with a set of simple action skills including navigating, grasping, recognizing objects or people, among others. The robot, using these skills, must complete a complex task (defined as the concatenation of several of those basic skills) encoded as a voice command in natural language. Thanks to the implemented architecture the robot is able to complete the task. *Soar* acts as the reasoner that selects the current action the robot must do, moving it towards the goal.

This system has been tested on the human size humanoid robot REEM in a home environment. The results support that the implemented architecture is capable of solving complex tasks such as "bring me a drink". In addition, this project also demonstrates the changeability of the architecture and its property to be installed on any robot regardless of its hardware.

Índex

Índex de figures

Índex de taules

Índex d'algorismes

Capítol 1

Contextualització del projecte

1.1 Informació, context i descripció del treball

Aquest treball de final de grau forma part del projecte REEM@IRI, una iniciativa formada per l'empresa PAL Robotics amb la col·laboració de l'Institut de Robòtica i Informàtica Industrial (IRI) de la Universitat Politècnica de Catalunya (UPC) i l'associació AESS Estudiants. L'objectiu d'aquesta iniciativa és formar un equip amb el propòsit de participar en la competició de robòtica RoboCup amb el robot humanoide REEM[1].

La RoboCup[2] és un projecte fundat amb el propòsit de promoure, mitjançant una competició anual, l'educació i la investigació de temes referents a la intel·ligència artificial i la robòtica. D'entre les diferents competències en les quals està dividida la competició, la intenció del projecte REEM@IRI és participar en la Robocup@Home[3]. Aquesta té com a objectiu promoure el desenvolupament de robòtica humanoide de servei per a futures aplicacions domèstiques mitjançant tot un seguit de proves de *benchmark*. Aquests desafiaments permeten avaluar el rendiment i les capacitats dels robots participants en un entorn familiar realista no estandarditzat.

Aquest treball final de grau està definit amb el propòsit d'implementar un nou mètode per resoldre una de les proves de la RoboCup@Home anomenada *Enduring General Purpose Service Robot*. Aquesta prova avalua les capacitats del robot de representar i raonar sobre l'escenari; interpretar, planificar i executar una tasca complexa; i reconèixer la parla. El robot en aquesta prova ha de completar una tasca seleccionada de forma aleatòria d'una gran llista predefinida. L'ordre seleccionada és dita al robot en llenguatge natural per un àrbitre. Dos exemples reals

[1] pal-robotics.com/en/robots/REEM
[2] www.robocup2014.org
[3] www.robocupathome.org

de possibles tasques demanades són "Bring me a drink"o "Go to reception, find a person and introduce yourself".

La gran dificultat del problema plantejat resideix principalment en dos factors. El primer és el fet que l'ordre és expressada en llenguatge natural amb informació mancant necessària per poder resoldre la prova. Per tant, el robot ha de poder identificar la informació mancant, demanar-la i entendre la resposta. El segon factor és la gran quantitat de possibles tasques diferents que el robot ha de poder resoldre.

Allunyant-nos del món de la competició, en un futur no molt llunyà la idea de robots humanoides de servei ajudant a les cases deixarà de ser un concepte de ciència-ficció per passar a ésser una realitat. En aquest futur escenari, és clar, que els robots hauran de poder entendre i completar una gran quantitat de possibles tasques resolent tots els possibles entrebancs que puguin trobar-se. Equipats amb la nova arquitectura que es plantejarà en aquest treball final de grau podrien aconseguir-ho.

En conclusió, aquest treball final de grau està dirigit a resoldre una de les proves de la RoboCup@Home, i els membres de l'equip REEM@IRI seran qui l'utilitzin i els seus principals beneficiaris. Tot i així, com ja s'ha vist, aquesta arquitectura podria ser utilitzada per empreses productores de futurs robots de servei. En aquest cas, els beneficiaris serien aquestes empreses, les quals podrien augmentar la intel·ligència dels seus robots; i les persones que comprin aquests robots, en tenir a la seva disposició robots capaços de resoldre un major nombre de tasques complexes.

1.2 Abast

Aquest treball final de grau definirà i implementarà una arquitectura cognitiva basada en Soar[4] per la resolució de tasques complexes suposant un robot de servei dotat de les habilitats descrites a la Figura 1.1 com desplaçar-se, reconèixer objectes, etc. El robot haurà de completar una tasca complexa dita en llenguatge natural que requereix aquestes habilitats per a ser resolta. El raonador Soar[5] haurà de seleccionar en cada moment l'acció a ser executada pel robot per a completar la tasca. La implementació es farà sobre l'entorn ROS[6] (Robot Operating System), el sistema operatiu estàndard de robots actualment. Aquest sistema serà provat en el robot humanoide de servei REEM.

Es partirà del supòsit que el robot en el qual se li apliqués aquesta arquitectura

[4] sitemaker.umich.edu/soar/home
[5] sitemaker.umich.edu/Soar/home
[6] www.ros.org

ja tindria implementat tot el conjunt d'habilitats requerides per a la resolució de totes les possibles tasques demanades, com per exemple, navegació i reconeixement de persones. En conseqüència, no s'entrarà en aquests temes.

També s'ha definit un extens conjunt específic de goals possibles que l'arquitectura ha de poder resoldre (veure Apèndix B i C). Aquesta, de fet, és la llista oficial de goals demanats en la competició Robocup@Home. Intenta reproduir les tasques més comunes que se li poden encarregar a un robot de servei en un ambient domèstic. Aquests goals estan definits per una acció que el robot ha de realitzar sobre un únic objecte determinat ("grasp a coke"), una única localització específica ("navigate to the kitchen table") i/o una única persona concreta ("bring me a drink").

Originalment, la resolució del problema plantejat en aquest treball es va orientar des de l'equip REEM@IRI com un projecte de col·laboració entre un company de l'equip i jo. Dintre de REEM@IRI, tota la part del parsing de veu serà realitzada pel meu company i l'autor d'aquest treball estarà encarregat de la resta. Tot i així, en aquest treball final de grau tota la part que realitzarà el meu company serà refeta per mi amb llibreries i plantejament completament diferents. Així doncs, tot el software entregat al final del treball haurà estat realitzat per mi.

HABILITATS	
Habilitat	Descripció
Go to	Anar a un lloc
Introduce himself	Parlar sobre si mateix
Follow person	Seguir la persona enfront del robot
Search objects	Buscar objectes enfront del robot
Search person	Buscar una persona per la zona actual
Grasp object	Agafar un objecte
Deliver object	Lliurar un objecte a la persona o localitat enfront del robot
Memorize person	Memoritzar persona enfront del robot
Exit apartment	Sortir de la casa
Recognize person	Reconèixer persona enfront del robot
Point at an object	Apuntar a un objecte

Taula 1.1: Taula de descripció d'habilitats implementades en el raonador

1.2.1 Objectius

Els objectius d'aquest treball són la definició i implementació d'una arquitectura cognitiva basada en Soar que compleixi les següents característiques:

- Ser capaç de reconèixer satisfactòriament la parla (ASR[7]) i fer una posterior anàlisi gramatical de la frase (parsing). Aquesta anàlisi ha de permetre identificar l'acció demanada a realitzar. També ha de retornar informació de les localitzacions, persones i/o objectes mencionats en l'ordre.
- Ser capaç de reconèixer informació no disponible necessària per la correcta realització de la tasca demanada. Un cop és identificada, demanar-la.
- Ser capaç de planificar la seqüència d'habilitats a executar per tal de resoldre la tasca requerida.
- Poder resoldre de forma no aleatòria quina és la següent acció a executar en un punt on l'arquitectura cognitiva no té suficient informació per decidir-la.
- Poder executar les habilitats del robot i recollir una realimneació d'aquestes.
- Ser eficient i ràpida però també altament canviable. Ha de ser capaç de poder utilitzar una nova habilitat només definint-la, sense que sigui necessari canviar la resta de l'arquitectura.
- Ser capaç de treballar de forma agnòstica al robot, és a dir, aquesta arquitectura ha de poder funcionar en qualsevol robot.

1.2.2 Obstacles i riscs

Durant el procés de realització d'aquest treball hom pot trobar-se amb diferents obstacles per causes derivades de o externes al treball, així com els riscos que comporten.

D'una banda, és possible que apareguin un conjunt d'**obstacles derivats** de les eines utilitzades al llarg del treball que suposin un risc pel correcte funcionament del programari desenvolupat. Un exemple d'aquests seria mancances en el programari de simulació que no permetin certes funcionalitats que el robot real sí tingui. Aquest obstacle en concret suposaria el risc de no poder fer una bona fase de test i, com a conseqüència, tenir un codi amb possibles errors. En cas de trobar aquest tipus d'obstacles es realitzarien programes de test exclusius propis per completar aquestes possibles mancances (mocks).

D'altra banda, també podrien aparèixer **obstacles externs** al treball que suposin un risc per a la correcta realització d'aquest. Aquests obstacles principalment

[7] Automatic Speech Recognition

vindrien donats per la no disponibilitat de recursos necessaris, com el robot REEM. Aquest obstacle en concret podria suposar el risc de perdre l'oportunitat de poder introduir aquesta arquitectura en un robot real. Per solucionar aquest tipus d'obstacles es buscarien alternatives, com per exemple, altres robots.

L'últim gran risc que comporta aquest projecte és el fet que s'implementarà amb un **entorn de programació desconegut** per mi, ROS. Treballar en un framework desconegut suposa un gran risc en tant que no s'està familiaritzat amb la forma d'interactuar amb ell i, per tant, suposa no tenir una imatge clara dels passos a seguir en la implementació ni el temps que comportaran. Aquest fet podria traduir-se a una **mala planificació temporal i de costos**.

1.3 Competències tècniques

En aquest apartat es justifica el desenvolupament de les competències tècniques associades al treball. També s'especifica el nivell de profunditat treballat de cada competències i les parts on s'ha tractat.

- **CCO1.1: Avaluar la complexitat computacional d'un problema, conèixer estratègies algorísmiques que puguin dur a la seva resolució, i recomanar, desenvolupar i implementar la que garanteixi el millor rendiment d'acord amb els requisits establerts.** [**Bastant**] Al llarg del treball s'han avaluat els diferents mòduls a implementar i s'han adaptat les llibreries, algoritmes i llenguatges de programació utilitzats d'acord a paràmetres d'eficiència, robustesa i canviabilitat. A més a més, s'ha millorat l'eficiència dels algoritmes implementant-los sobre un framework específic per a robots amb el llenguatge més adient per cada cas (compilats o interpretats). Aquesta competència s'ha treballat en tots els mòduls implementats.

- **CCO2.1: Demostrar coneixement dels fonaments, dels paradigmes i de les tècniques pròpies dels sistemes intel·ligents, i analitzar, dissenyar i construir sistemes, serveis i aplicacions informàtiques que utilitzin aquestes tècniques en qualsevol àmbit d'aplicació.** [**En profunditat**] Aquesta competència s'ha treballat sobretot en els mòduls *Extractor semàntic* i *Raonador*. En el mòdul *Extractor semàntic* s'han aplicat paradigmes d'*aprenentatge automàtic* per entrenar el parser. El submòdul *Soar* està implementat sobre una *arquitectura cognitiva* on se li han definit totes les habilitats del robot. En aquest submòdul també s'han aplicat tècniques pròpies dels sistemes intel·ligents com *Sub-goal capacity* i *Chunking ability*.

- **CCO2.2: Capacitat per a adquirir, obtenir, formalitzar i represen-**

tar el coneixement humà d'una forma computable per a la resolució de problemes mitjançant un sistema informàtic en qualsevol àmbit d'aplicació, particularment en els que estan relacionats amb aspectes de computació, percepció i actuació en ambients o entorns intel·ligents. [**Bastant**] Aquesta competència s'ha treballat molt en desenvolupar software a ser integrat en un robot. El software implementat utilitza coneixement extret dels sensors del robot i defineix una representació del món a partir d'aquesta informació. Aquesta competència s'ha treballat sobretot en el mòdul *Raonador*.

- **CCO3.1: Implementar codi crític seguint criteris de temps d'execució, eficiència i seguretat.** [**Bastant**] Tot el codi implementat en aquest treball és altament eficient. Es pot comprovar l'eficiència del codi en el vídeo presentat de la demo "bring me a drink". En aquest vídeo s'observa l'alta velocitat d'execució de l'arquitectura. Aquesta competència s'ha treballat en tots els mòduls implementats.

1.4 Estat de l'art

Des de les primeres implementacions de robots fa més de 50 anys, la robòtica ha estat aplicada principalment en sistemes industrials amb el propòsit d'augmentar la producció. Els robots utilitzats eren els coneguts *robots industrials*.

En les últimes dècades ha aparegut la necessitat i voluntat de poder aplicar la robòtica en altres camps. Com a conseqüència, a finals dels anys 80 va aparèixer el terme *robots de servei*. Aquests robots és caracteritzen per treballar en entorns no controlats canviants i per tenir una forta interacció amb les persones.

Segons la *Federació Internacional de Robòtica*[8] (IFR), un robot de servei és un robot que opera de manera parcial o totalment autònoma, per realitzar serveis útils per al benestar dels humans i de l'equipament, excloent operacions de manufactura.

En altres paraules, els robots de servei són la implementació moderna d'un dels grans somnis de la humanitat que és tenir robots personals/assistents.

Tal com ja s'ha descrit, els robots de servei comparteixen l'espai de treball amb les persones i interactuen amb elles. És clar doncs, que un dels factors imprescindibles que es té en compte és la interacció robot-persona (HRI)[26, 17]. HRI és l'estudi de la interacció entre robots i persones que intenta definir la forma d'interacció física i social més correcte entre ells. En aquest treball la interacció robot-persona es farà mitjançant la parla amb l'objectiu de ser el més intuïtiva i fàcil possible per a les persones.

[8] www.ifr.org

És important remarcar que l'arquitectura que s'implementarà en aquest treball és la primera arquitectura cognitiva per a la resolució de tasques complexes en robòtica de servei que contempla tot el procés des que s'escolta l'ordre fins a la finalització de la tasca. Per tant, no hi ha productes similars o relacionats. En els següents punts es realitza una anàlisi de l'estat de l'art dels diferents components que formaran part d'aquesta arquitectura.

1.4.1 Reconeixement automàtic de la parla

El *reconeixement automàtic de la parla* (ASR) [31] és una disciplina de la intel·ligència artificial que consisteix a traduir la parla a text. Durant els últims anys ha experimentat un gran creixement gràcies a la introducció d'aquesta tecnologia en el sector de la telefonia.

Actualment, hi ha un gran nombre de ASRs disponibles. Alguns d'aquests són APIs públiques, com la *Google's Web Speech API*[9] i la *Microsoft Speech API*[10]. També hi ha un gran nombre de programari open source com *Sphinx* [32] i *Julius* [1]. Per contra, també existeix una llarga llista de programari privatiu, com *Nuance Dragon*[11].

Els sistemes de reconeixement de veu de propòsit general moderns es basen en *Models ocults de Markov* (HMM)[14, 22]. Aquests són models estadístics que de sortida retornen una seqüència de símbols o quantitats. Els HMM s'utilitzen en el reconeixement de veu, ja que la veu pot ser tractada com un senyal definit a trossos o un senyal estacionari de curta durada. Altres raons que fan els HMM populars són la possibilitat de ser entrenats de forma automàtica, la seva simplicitat i computacionalment factibles d'utilitzar.

Històricament també s'ha aplicat l'algoritme 'time warping' per fer reconeixement de veu [12]. Aquest algorisme permet mesurar la similitud entre dues seqüències que poden variar en temps o velocitat. Tot i així, aquest sistema ha estat desplaçat per l'enfocament basat en models ocults de Markov.

Per últim, a finals de 1980 va sorgir la idea d'utilitzar xarxes neuronals per reconèixer la parla [23]. Aquest enfocament té una gran taxa de reconeixements amb èxit en unitats curtes de temps com fonemes individuals i paraules soles. Tot i així, rarament té èxit reconeixent frases llargues a causa de la seva dificultat de modelar dependències temporals. A diferència dels HMM, les xarxes neuronals no fan suposicions sobre característiques estadístiques distintives. Actualment les xarxes neuronals en ASRs només s'utilitzen per fer preprocessament de les dades, per exemple, s'utilitzen per fer reducció de la dimensió d'aquestes [9].

[9] www.goo.gl/lA0Pfd
[10] www.goo.gl/aW0ULH
[11] www.nuance.es/dragon/index.htm

Es va consultar a l'expert Matthew Walter[12] del MIT[13] quina tecnologia utilitzar per introduir l'ASR en l'arquitectura i va recomanar la llibreria **Google speech recognition**. Els principals avantatges que ofereix aquesta llibreria són la seva fàcil integració, utilització i alta fiabilitat de resultats. Tot i així, aquesta, requereix accés a Internet per funcionar. Aquest fet no resulta cap inconvenient per l'arquitectura que s'implementarà, ja que els robots en els quals s'introduiria es mouen en entorns que disposen de punts d'accés a Internet.

1.4.2 Comprensió del llenguatge natural

La *comprensió del llenguatge natural* (NLU) [30] és una disciplina de la intel·ligència artificial que s'ocupa de la comprensió lectora d'una màquina. De forma equivalent als ASRs, durant els últims anys la comprensió del llenguatge natural ha experimentat un gran creixement gràcies a la introducció d'aquesta tecnologia en el sector de la telefonia.

Moltes estratègies i enfocaments diferents han estat realitzats per intentar resoldre aquest problema. Un gran nombre d'aquestes estratègies es discuteixen a [4].

Una de les solucions més comunes és fer ús d'ontologies de dependència semàntica com WordNet [25] o corpus com VerbNet [28]. Una altra solució molt comuna consisteix en aplicar principis d'aprenentatge automàtic (*machine learning*). Aquesta solució planteja entrenar un model capaç d'estructurar el text introduït a partir d'exemples de texts ja estructurats [29]. Per altra banda, hi ha altres enfocaments completament diferents que fan ús de la informació sensorial del robot per a poder realitzar una millor comprensió [37, 3].

Per introduir NLU en l'arquitectura s'utilitzarà la tecnologia **h2sl**. Aquest paquet proporciona el codi font d'un conjunt de llibreries i executables que converteixen l'entrada de text de forma lliure a una instància de llenguatge estructurat. Aquest paquet està basat en grafs distribuïts de correspondència [10]. *h2sl* també ha estat recomanat per Matthew Walter per la seva potència i fàcil comunicació amb les llibreries *Google speech recognition*. A més a més, aquestes llibreries ofereixen una interfície gràfica per visualitzar els resultats que facilitarà la fase de test i la comprensió per part de futurs usuaris de l'arquitectura.

1.4.3 Sistemes de raonament

Els enfocaments típics per al control general dels robots de servei es basen principalment en màquines d'estat, on tots els passos necessaris per aconseguir completar

[12] people.csail.mit.edu/mwalter/
[13] Massachusetts Institute of Technology

una tasca són especificats i coneguts prèviament pel robot. En aquests controladors, es crea de forma exhaustiva la llista d'accions possibles que el robot pot fer, així com tots els passos necessaris per executar-les. El problema amb aquest enfocament és que tot ha de ser prèviament especificat, impedint així que el robot pugui respondre davant de noves situacions o de noves tasques.

Una alternativa a les màquines d'estat és l'ús de planificadors [36]. Els planificadors decideixen en temps real la millor seqüència d'habilitats a ser executades per tal d'assolir l'objectiu, generalment amb enfocaments probabilístics. Un enfocament diferent de l'ús de planificadors és l'ús d'arquitectures cognitives. Aquests sistemes intenten imitar alguns dels processos del cervell per tal de generar una decisió [33, 2, 13, 15, 21, 19, 7].

Hi ha diverses arquitectures cognitives disponibles: Soar [20], ACT-R[14] [2, 35], CRAM[15] [5], SS-RICS[16] [15]. De tots elles, només CRAM i SS-RICS han estat dissenyades específicament per a ser aplicades en robots. CRAM i SS-RICS ja han estat utilitzats per a la resolució de tasques complexes, com per exemple, cuinar pastissos [6]. Per altra banda, tot i que Soar no està pensat per ser aplicat en robots, ha estat utilitzat per implementar tasques molt senzilles de navegació [8]. A la Taula 1.2 es realitza una comparativa més detallada.

Es pot veure que, de tots elles, només CRAM i SS-RICS han estat dissenyades específicament per a ser aplicades en robòtica. Tot i així, CRAM només és capaç de construir i executar plans predissenyats, és a dir, CRAM és incapaç de resoldre noves situacions no esperades. Per tant, limita les accions que el robot pot executar a aquelles que han estat predefinides. Per aquests motius, l'arquitectura CRAM ha estat descartada. Per altra banda, SS-RICS ha estat descartat per la seva gran complexitat en tenir poc temps per finalitzar aquest treball.

Finalment, descartades les dues arquitectures orientades a robots, s'ha escollit utilitzar l'arquitectura Soar, i no ACT-R, en aquest treball. Aquesta decisió s'ha

[14] Adaptive Control of Thought—Rational
[15] Cognitive Robotic Abstract Machine
[16] Symbolic and Subsymbolic Robotic Intelligence Control System

	Orientat a	Raonament simbòlic	Control de baix nivell	Complexitat de la programació
Soar	Cognició	Complet	Complex	Simple
CRAM	Robot	Simple	Integrat	Complex
ACT-R	Cognició	Complet	Complex	Complex
SS-RICS	Robot	Simple	Integrat	Complex

Taula 1.2: Taula comparativa de diverses arquitectures cognitives.

pres pel fet que Soar és una arquitectura cognitiva general capaç de seleccionar l'acció requerida per a l'actual situació i meta sense tenir una llista predefinida de plans o situacions. A més a més, no té una gran complexitat en comparació a l'arquitectura ACT-R.

1.5 Recursos

Aquest treball farà ús d'un conjunt de recursos hardware, software i humans.

Hardware

- REEM, el robot humanoide de servei creat per l'empresa PAL Robotics. (Figura 1.2)
- Asus Xtion acoblada al robot per poder fer detecció d'objectes i persones.
- Ordinador Pavilion dv6.

Software

- Ubuntu 12.04
- Els editors de text Sublime Text i Vim.
- El framework ROS (Robot Operating System) pel desenvolupament del software.
- El simulador multi-robot per entorns interiors i exteriors anomenat Gazebo. (Figura 1.1a)
- L'eina Rviz per la visualització 3D del entorn. Mostra el que el robot està veient, pensant i fent. (Figura 1.1b)
- L'arquitectura cognitiva Soar.
- Tot el software integrat en el robot REEM, per exemple, reconeixedor d'objectes i persones; mòduls de navegació, etc.
- ShareLaTeX, un editor online de LaTeX.
- El software de control de versions Subversion.
- El programari de gestió de referències Mendeley.

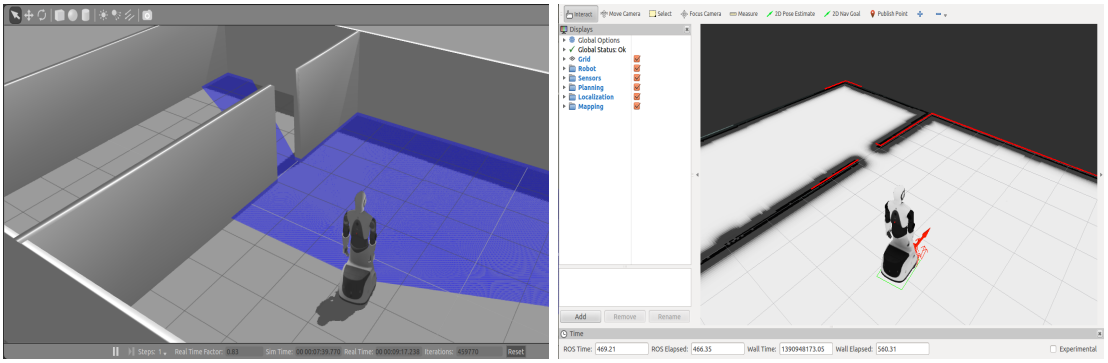
Humans Membres del projecte REEM@IRI experts en els camps de la robòtica i la intel·ligència artificial. Aquest conjunt de professionals aportaran feedback al llarg del projecte.

1.5.1 Sistema operatiu robòtic (ROS)

ROS és l'actual sistema operatiu estàndard per robots. Aquest proporciona eines que faciliten molt la implementació de software per robots. Entre les diferents eines que proporciona destaquen: abstracció de maquinari, control de dispositius de baix nivell, implementació de funcionalitats d'ús comú, pas de missatges entre processos i gestió de paquets. A més a més, també proporciona eines per visualitzar i depurar el codi.

ROS està organitzat com una xarxa de nodes i arestes. Cada node és un executable el qual està connectat amb altres nodes. Qualsevol node del sistema es pot connectar a qualsevol node, veure la informació que està enviant i publicar informació a la xarxa. Els dos conceptes necessaris per comprendre la implementació d'aquest treball són: *nodes*, *messages* i *topics*.

- **Node:** Un *node* és un executable que utilitza ROS per comunicar-se amb altres *nodes*.
- **Message:** Els *nodes* es comuniquen entre ells mitjançant *messages*. Cada missatge conté la informació que es vol enviar a altres *nodes*.
- **Topic:** Els *topics*, en una xarxa, representarien les arestes que connecten els nodes entre si. Els *topics* són canals pels quals s'envien els *missages*. Els *nodes* poden publicar missatges en un *topic*, així com subscriure's a un per rebre missatges.



(a) Simulador Gazebo (b) Visor Rviz

Figura 1.1: Algunes de les eines utilitzades

Figura 1.2: Robot REEM

- **Service:** Els *serveices* són una altra manera que els nodes es poden comunicar entre si. Aquests permeten als *nodes* enviar sol·licituds i rebre una resposta.

1.5.2 Robot REEM

El robot utilitzat per provar el sistema desenvolupat és el robot REEM, un robot humanoide de servei creat per l'empresa PAL Robotics. El seu pes és d'aproximadament 90 kg, té 22 graus de llibertat i una autonomia d'aproximadament 8 hores. Aquest robot utilitza OROCOS per a les operacions en temps real i ROS per la implementació d'habilitats. Entre altres habilitats, pot reconèixer i comprendre objectes, detectar rostres i seguir una persona. El robot està equipat amb un processador Core 2 Duo i un ordinador ATOM, que proporcionen tota la potència de càlcul necessària per dur a terme tot el control. Això vol dir que tots els algorismes necessaris per planificar i dur a terme totes les habilitats s'executen dintre del robot. A més a més, per tal de poder fer una millor detecció d'objectes i persones s'ha afegit un sensor Kinect a la versió comercial.

1.6 Identificació de lleis i regulacions

Aquest treball final de grau fa ús d'una gran quantitat de fonts i recursos. Aquests estan coberts per un seguit de llicències que afecten directament el treball i s'hau-

ran de tenir en compte. En aquest apartat es descriu el conjunt de lleis i regulacions que cobreixen cadascuna de les fonts i recursos d'aquest treball.

Aquest treball complirà tot aquest conjunt de lleis i regulacions.

1.6.1 Anàlisi de les lleis i regulacions de les fonts i recursos

REEM@IRI Tal com s'ha descrit anteriorment, aquest treball forma part del projecte REEM@IRI. Per tant, les lleis i regulacions del projecte REEM@IRI afecten directament a aquest treball. Aquestes defineixen que tot el codi del projecte REEM@IRI és privat i només poden tenir-hi accés els seus membres.

Pal Robotics Aquest treball farà ús del robot REEM i del programari que hi és integrat. Tot aquest software és de l'empresa Pal Robotics i és propietari. Tot i així, hi ha un conjunt de paquets de codi públic per a ROS sota les llicències *Creative Commons BY-NC-ND 3.0* i *BSD*.

IRI Aquest treball també farà ús de llibreries pròpies de l'Institut de Robòtica i Informàtica Industrial (IRI). Aquestes llibreries estan cobertes per la llicència *GNU Lesser General Public License*.

Soar Soar està protegit per la llicència *BSD*.

Google speech recognition Aquest treball també farà ús de les llibreries de *Google speech recognition*. Aquestes llibreries estan lligades a la llicència *MIT License*.

h2sl L'arquitectura implementada utilitzarà la llibreria *h2sl* per introduir el NLU. Aquesta està coberta per la llicència *GNU Lesser General Public*.

ROS Tal com s'ha detallat anteriorment, aquest treball es desenvoluparà sobre el framework ROS. El conjunt de llibreries que formen ROS estan cobertes per les llicències *BSD*[17] *modificada* i *Apache License 2.0*.

Robòtica de servei La robòtica de servei és estandarditzada pel comitè *ISO/TC 184/SC 2 Robots and robotic devices*. Aquest comitè defineix temes com vocabulari, requeriments de seguretat en entorns robotitzats i mètodes de test.

[17] Berkeley Software Distribution

1.6.2 Descripció de les llicències identificades

Creative Commons BY-NC-ND 3.0 License

Creative Commons és una organització sense ànim de lucre que ofereix a autors i creadors poder compartir voluntàriament el seu treball, lliurant llicències i eines lliures. Aquestes llicències defineixen l'ús i benefici que en poden fer els usuaris i els drets que es reserven els autors.

La llicència *Creative Commons BY-NC-ND 3.0*[18] estableix el següent:

- **Compartir**: Es pot copiar i redistribuir el material en qualsevol mitjà o format.
- **Reconeixement**. S'ha de reconèixer adequadament l'autoria, proporcionar un enllaç a la llicència i indicar si s'han fet canvis. Pot fer-ho de qualsevol manera raonable, però no d'una manera que suggereixi que té el suport del llicenciador o el rep per l'ús que fa.
- **No Comercial**: No es pot utilitzar el material per a una finalitat comercial.
- **Sense ObraD erivada**: Si es remescla, transforma o crea material a partir del material cobert, no es pot difondre.

Llicència BSD

La llicència *BSD* pertany al grup de llicències de programari lliure que imposen restriccions mínimes sobre la redistribució de programari cobert. La llicència obliga als autors que el codi font i els binaris que distribueixin han de tenir escrit el copyright i les condicions de la llicència. Per altra banda, estableix que l'usuari té dret a utilitzar lliurement el codi sense necessitat de mantenir la llicència.

Llicència BSD modificada

Ja s'ha explicat anteriorment les característiques de la llicència *BSD*. Tot i així, la llicència que utilitza ROS és una versió de BSD modificada, la *BSD 3-Clause License*[19].

En el cas de ROS, la llicència fixa que és obligatori indicar que s'utilitza software desenvolupat a la Universitat de Califòrnia en la publicitat que faci referència a material que utilitza contingut cobert per aquesta llicència.

[18] creativecommons.org/licenses/by-nc-nd/3.0/
[19] opensource.org/licenses/BSD-3-Clause

GNU Lesser General Public License

Aquesta llicència va ser creada per la *Free Software Foundation*, una organització fundada amb el propòsit d'impulsar el programari lliure i en particular el projecte GNU.

La *GNU Lesser General Public License*[20] permet la llibertat de compartir i modificar el programari cobert per ella, assegurant que el programari és lliure per a tots els usuaris. Altres activitats que no siguin còpia, distribució o modificació no estan cobertes en aquesta llicència i estan fora del seu abast.

Apache License 2.0

Aquesta llicència va ser creada per la *Apache Software Foundation* (ASF) fundada amb l'objectiu de donar suport a projectes Apache.

La *Apache License 2.0*[21] permet a l'usuari usar el programari per a qualsevol propòsit, distribuir-lo, modificar-lo, i generar versions modificades d'aquest. La llicència no exigeix que el material derivat es distribueixi utilitzant la mateixa llicència, ni tan sols que s'hagi de distribuir com a programari lliure. Només exigeix que es notifiqui als receptors del nou programari que s'ha fet servir codi protegit amb la llicència Apache.

MIT License

La *MIT License*[22] concedeix permís, de forma gratuïta, a qualsevol persona que obtingui una còpia d'aquest programari i arxius de documentació associats a aquest. Atorga el dret d'utilitzar, copiar, modificar, fusionar, publicar, distribuir, sublicenciar i / o vendre còpies del programari cobert sense limitacions.

[20] opensource.org/licenses/lgpl-license
[21] www.apache.org/licenses/LICENSE-2.0.html
[22] opensource.org/licenses/MIT

Capítol 2

Metodologia de treball

La metodologia que s'aplicarà en aquest treball serà una metodologia àgil [24]. Actualment aquestes metodologies són molt utilitzades i companyies com IBM[1], Amazon i Google les apliquen. Estan basades en el desenvolupament iteratiu i incremental. Així doncs, el treball es realitzarà mitjançant un seguit d'iteracions. Cada una d'elles està dividida en anàlisi, implementació, test, integració i documentació. (Veure Figura 2.1)

Aplicar metodologies àgils permetrà tenir una major proximitat amb el tutor així com una capacitat de resposta més alta envers a obstacles i demandes del tutor. Per altra banda, també reduirà el risc de tenir una mala planificació temporal i de costs. Això és gràcies al fet que les metodologies àgils permeten revisar i adaptar dinàmicament la planificació inicial.

A continuació es defineixen breument les quatre iteracions en les quals està dividit el treball, la definició completa es descriu a la Secció 2.3.1.

Set Up En la primera iteració es realitzarà un set-up del sistema així com un autoaprenentatge del diferent programari que s'utilitzarà al llarg del treball.

ASR La segona consistirà en desenvolupar el mòdul de ASR, encarregat del reconeixement de la parla. Aquest traduirà el senyal de veu de l'ordre a text.

Extractor semàntic A la tercera iteració s'implementarà l'extractor semàntic, encarregat de fer una anàlisi gramatical d'una frase, identificant així el rol de cada paraula en l'oració (verb, subjecte, etc).

[1]International Business Machines

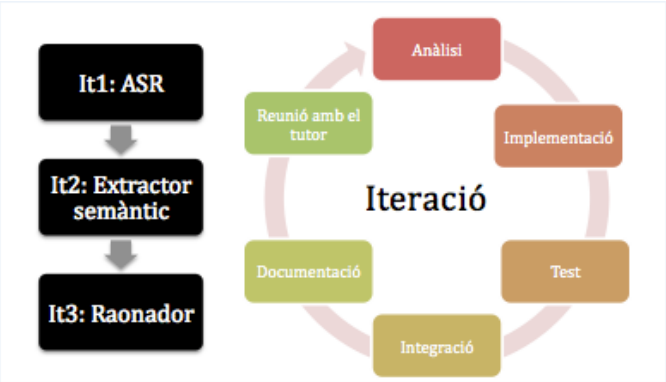


Figura 2.1: Diagrama metodologia àgil definida

Raonador Finalment, a l'última iteració es desenvoluparà el raonador, encarregat d'anar seleccionant les accions a executar al llarg de la realització de la tasca.

Al final de cada iteració es realitzarà una reunió amb el director per tal d'obtenir una realimentació que ajudi a una millor execució del treball.

2.1 Eines de seguiment

Al llarg del treball s'utilitzaran un conjunt d'eines per facilitar el seu seguiment.

En primer lloc, es farà ús d'eines per tal de poder mantenir una comunicació constant amb el tutor així com amb possibles terceres persones amb les quals fos necessari contactar. Amb aquest objectiu, es farà servir el **correu electrònic** i **Skype**.

Per l'emmagatzemat i la compartició d'arxius es farà ús del servei d'allotjament de fitxers en el núvol **Dropbox**. També s'utilitzarà **Mendeley**[2] com a programari de gestió de referències.

Finalment, s'utilitzarà **Subversion** com a software de control de versions amb l'objectiu de mantenir una còpia de seguretat de les diferents versions del programa.

2.2 Mètode de validació

El mètode de validació constarà de diferents fases. L'ordre en què estan definides és l'ordre en el qual han estat executades.

[2] www.mendeley.com

1. Comprovar el correcte funcionament del codi a la consola de l'ordinador.
2. Introduir el codi del programa en l'eina de simulació gazebo[3] proporcionada pel framework ROS. Aquest simulador ofereix un entorn virtual controlat de simulació amb condicions ideals.
3. Realitzar proves en el robot humanoide REEM en entorns controlats un cop el resultat és l'esperat.
4. Sotmetre el robot a proves en entorns reals no controlats.

2.3 Planificació temporal

En aquest capítol es defineixen les tasques a realitzar i la planificació temporal del treball.

2.3.1 Descripció de les tasques

Definició del projecte

La primera tasca és la definició del projecte que forma part de l'assignatura *Gestió de Projectes*. S'ha definit el treball segons els paràmetres: abast, planificació temporal, pressupost i sostenibilitat, i estat de l'art.

Iteracions del treball

Com ja s'ha comentat, aquest treball es desenvoluparà amb metodologies àgils. Així doncs, el treball es realitzarà mitjançant un seguit d'iteracions detallades a continuació. Cadascuna d'elles està dividida en anàlisi, implementació, test, integració i documentació.

Tal com s'ha definit en el mètode validació del treball (Secció 2.2), la part de test de cada iteració estarà formada per les fases de test en: consola de l'ordinador, simulador Gazebo i finalment en el robot humanoide de servei REEM.

L'ordre en què estan definides les iteracions és l'ordre en el qual seran executades.

Set Up inicial. Aquesta primera iteració consistirà en instal·lar i configurar tot el programari necessari per poder començar a desenvolupar el treball. També,

[3] wiki.ros.org/gazebo

es realitzarà un autoaprenentatge del framework ROS així com de les diferents llibreries que s'utilitzaran per a la implementació del treball.

ASR. Aquesta iteració té com a objectiu desenvolupar el mòdul ASR encarregat del reconeixement automàtic de la parla. Aquest mòdul, en l'arquitectura final, tindrà la funció de transformar les ones sonores de l'ordre pronunciada al robot per veu a paraules.

Extractor semàntic. La tercera iteració consistirà en implementar el mòdul encarregat de fer una anàlisi gramatical d'una frase per tal d'identificar el rol de cada paraula en l'oració (verb, subjecte, etc). El mòdul desenvolupat en aquesta iteració servirà per poder identificar l'acció demanada a realitzar en la comanda verbal al robot. També indicarà les localitzacions, persones i/o objectes mencionats en l'ordre.

Raonador. L'última iteració serà destinada a desenvolupar el sistema de raonament cognitiu. Aquest serà l'encarregat d'anar seleccionant les accions que el robot ha de realitzar per tal de completar la tasca demanada amb èxit.

Redacció final de la memòria

En aquesta tasca es realitzarà la documentació final del treball. Consistirà en agrupar tota la documentació elaborada fins al moment i acabar de perfilar-la.

Preparació de la presentació

En l'última fase del treball s'escriurà el guió i el material de suport de la presentació final. També es realitzaran un seguit d'assajos per tal de trobar possibles millores de la presentació i practicar el temps de duració d'aquesta.

2.3.2 Estimació del temps

A continuació es mostra una definició del temps estimat per completar les tasques definides.

ESTIMACIÓ DEL TEMPS PER TASCA	
Tasca	Temps [h]
Definició del projecte	80
It0: Set Up inicial	85
It1: ASR	100
It2: Extractor semàntic	110
It3: Raonador	150
Redacció final de la memòria	15
Preparació de la presentació	25
TOTAL	**565**

Taula 2.1: Taula d'estimació del temps per tasca.

2.3.3 Pla d'acció i valoració d'alternatives

Els motius que podrien provocar desviacions en la planificació inicial són els següents:

- En la fase de test s'arriba a la conclusió que una part de l'arquitectura desenvolupada ha de ser replantejada, i, per tant, refeta. Aquest entrebanc es traduiria a un consum major del temps previst per aquella fase, provocant així un endarreriment en la planificació inicial. També podria suposar un augment de la utilització dels recursos humans en haver d'anar a parlar amb professionals del sector per tal d'arribar a un nou enfocament del problema.
- El robot no està disponible. Aquest fet afectaria la fase de test de cadascuna de les iteracions, impossibilitant fer proves en el robot. En aquest cas, com a alternativa, les proves només es realitzarien en consola i simulació. No suposaria una despesa extra dels recursos.
- El robot no es comporta com s'espera. És molt comú que codi que funciona en simulació, en el robot, en canvi, no ho faci. En el robot un gran nombre de factors poden provocar el malfuncionament de l'algorisme, per exemple, algun dels sensors no funciona correctament, condicions adverses de llum o, fins i tot, diferències entre les configuracions del robot de simulació i el real. Aquestes adversitats podrien provocar un augment del temps de test, provocant així un retard en les tasques posteriors i una despesa extra del recurs *robot REEM*. Per minimitzar aquest risc, es farà sempre ús del programari

rviz[4] proveït pel framework ROS. Aquest software permet monitoritzar el que el robot està veient, processant i realitzant en temps real.

Les metodologies àgils permeten revisar i adaptar dinàmicament la planificació inicial. Així doncs, si les diferents fases esmentades en la Secció 2.3.1 tenen una durada diferent de l'estimada, es modificarà la planificació. En el cas que una tasca duri més de l'esperat provocarà un retard en les següents tasques. Per contra, si una tasca es completa amb un temps inferior a l'estipulat a la planificació, s'iniciarà a l'acte la següent. Per altra banda, si el motiu de la desviació ve donat per culpa d'algun recurs extern al treball, s'ometrà temporalment la tasca que requereixi aquell recurs o es buscaran alternatives al recurs mancant.

El total d'hores estimades requerides per completar aquest treball són 570 hores. Aquestes han estat calculades a partir de l'estimació total de la Secció 2.3.2 més les hores aproximades de reunions amb el tutor que es duran a terme en finalitzar cada tasca. La durada d'aquest treball és d'aproximadament 18 setmanes, així doncs, serà necessària una dedicació de 31.6 hores setmanals. Així, aquest treball i la seva planificació són assolibles.

2.3.4 Diagrama de Gantt

	Nombre	Inicio	Fin
1	**Definició del projecte**	**17/02/2014**	**14/03/2014**
2	Abast	17/02/2014	20/02/2014
3	Planificació Temporal	21/02/2014	21/02/2014
4	Pressupost i sostenibilitat	24/02/2014	27/02/2014
5	Presentació preliminar	28/02/2014	03/03/2014
6	Context i bibliografia	04/03/2014	07/03/2014
7	Document i presentació final; i plec de condicions	10/03/2014	14/03/2014
8	**It0: Set Up Inicial**	**17/03/2014**	**02/04/2014**
9	Instal·lar i configurar software	17/03/2014	19/03/2014
10	Autoaprenantatge	20/03/2014	02/04/2014
11	**It1: ASR**	**03/04/2014**	**24/04/2014**
12	Anàlisis	03/04/2014	04/04/2014
13	Implementació	07/04/2014	14/04/2014
14	Testing	15/04/2014	17/04/2014
15	Integració	18/04/2014	22/04/2014
16	Documentació	23/04/2014	24/04/2014
17	**It2: Extractor Semàntic**	**25/04/2014**	**16/05/2014**
18	Anàlisis	25/04/2014	28/04/2014
19	Implementació	29/04/2014	06/05/2014
20	Testing	07/05/2014	09/05/2014
21	Integració	12/05/2014	14/05/2014
22	Documentació	15/05/2014	16/05/2014
23	**Raonador**	**19/05/2014**	**18/06/2014**
24	Anàlisis	19/05/2014	20/05/2014
25	Implementació	21/05/2014	04/06/2014
26	Testing	05/06/2014	11/06/2014
27	Integració	12/06/2014	16/06/2014
28	Documentació	17/06/2014	18/06/2014
29	Redacció final de la memòria	19/06/2014	20/06/2014
30	Preparació de la presentació	23/06/2014	02/07/2014

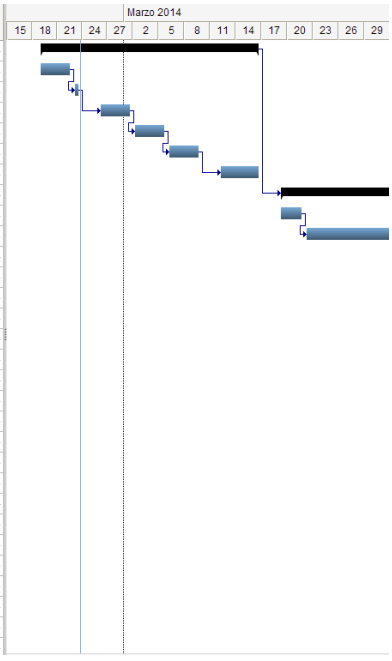


Figura 2.2: Diagrama de Gantt des de l'inici del treball a finals de març.

[4] wiki.ros.org/rviz

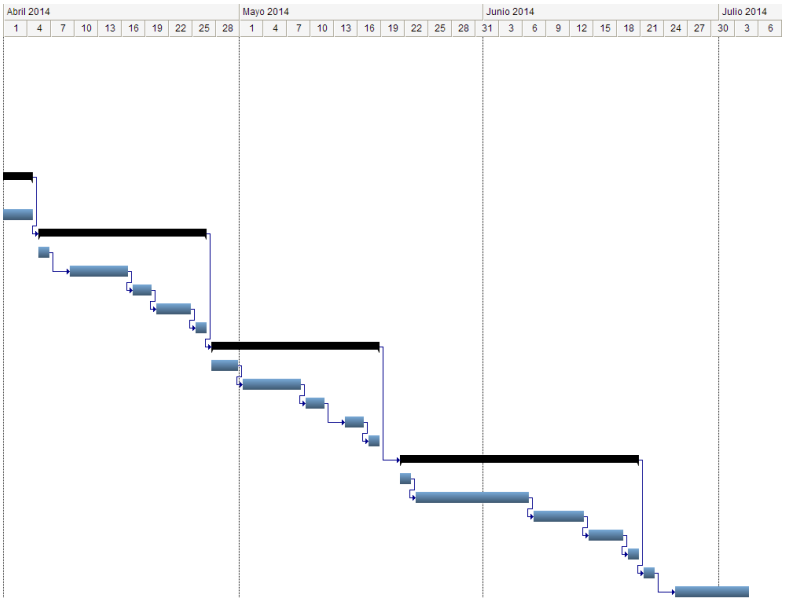


Figura 2.3: Diagrama de Gantt des d'inicis d'abril fins a la finalització del treball.

2.3.5 Planificació final

Com es pot observar en el diagrama de Gantt de la planificació inicial, en el moment de la fita de seguiment (24/04/2014), s'hauria d'haver estat realitzant la fase de test de la iteració *ASR*. Tot i així, s'han dedicat més hores setmanals que les establides inicialment i en aquell punt el treball (30/05/2014) estava en la fase d'integració de la iteració *Extractor semàntic*. Per tant, aquest treball anava un mes avançat respecte a la planificació inicial.

Tal com es va indicar inicialment, la metodologia d'aquest treball està basada en metodologies àgils. Aquestes permeten revisar i adaptar dinàmicament la planificació inicial. Per aquest motiu, es va adaptar la planificació inicial tenint en compte l'estat del treball en aquell punt. A continuació es mostra el diagrama de Gantt de la nova planificació.

Nombre	Inicio	Fin
It2: Extractor Semàntic	**24/04/2014**	**30/04/2014**
Integració	24/04/2014	28/04/2014
Documentació	29/04/2014	30/04/2014
Raonador	**01/05/2014**	**09/06/2014**
Anàlisis	01/05/2014	02/05/2014
Implementació	05/05/2014	19/05/2014
Testing	20/05/2014	26/05/2014
Integració	27/05/2014	04/06/2014
Documentació	05/06/2014	09/06/2014
Redacció final de la memòria	10/06/2014	18/06/2014
Preparació de la presentació	19/06/2014	03/07/2014

Figura 2.4: Diagrama de Gantt de la planificació que finalment s'ha seguit.

Aquesta nova planificació atorga més temps del definit inicialment en les fases més crítiques restants. Per exemple, anteriorment es disposava de 23 dies per realitzar la iteració *Raonador*, i es va canviar a 28. A més a més, també es va ampliar el nombre de dies per preparar el guió i el material de suport per la lectura; i per finalitzar la documentació del treball.

Aquests canvis respecte a la planificació inicial no afecten la definició d'objectius.

2.4 Metodologia final

No s'ha produït cap canvi respecte a la metodologia proposada inicialment, ja que aquesta ha permès tenir una major proximitat amb el tutor així com una capacitat de resposta més alta envers a obstacles i imprevistos. Per altra banda, també ha ajudat a reduir el risc de tenir una incorrecta planificació temporal i de costs inicial gràcies al fet que les metodologies àgils permeten revisar i adaptar dinàmicament la planificació inicial.

Tampoc s'ha canviat el mètode de validació. S'han aplicat totes les fases que es van proposar.

Capítol 3

Pressupost

Tal com s'ha detallat en la Secció 1.5, sobre recursos del treball, per dur a terme aquest projecte es farà ús d'un seguit de recursos hardware, software i humans. Part d'aquests recursos tenen un cost. En aquest capítol es farà una estimació d'aquests costs així com del cost de les despeses generals del treball. Per calcular el cost de software i hardware es tindrà en compte la seva amortització. Per altra banda, per calcular el cost dels recursos humans es considerarà la càrrega de treball definida en la planificació (31.6 hores setmanals).

3.1 Estimació dels costos

Els recursos hardware i software es faran servir de forma constant al llarg de tot el desenvolupament del treball. En conseqüència, aquests recursos no s'han dividit en cadascuna de les diferents tasques que es van definir en la planificació.

3.1.1 Hardware

COST HARDWARE				
Producte	Preu	Unitats	Vida Util	Amortització
Robot REEM*	0 €	1	4 anys	0 €
Asus Xtion*	0 €	1	3 anys	0 €
Ordinador Pavilion dv6	999 €	1	4 anys	70.55 €
TOTAL	**999 €**			**70.55 €**

Taula 3.1: Taula de descripció del cost del hardware.

3.1.2 Software

COST SOFTWARE				
Producte	Preu	Unitats	Vida Util	Amortització
Ubuntu 12.04	0 €	1	1 any	0 €
Sublime Text	0 €	1	1 any	0 €
Vim	0 €	1	1 any	0 €
ROS	0 €	1	1 any	0 €
Gazebo	0 €	1	1 any	0 €
Soar	0 €	1	2 anys	0 €
Software robot REEM*	0 €	1	0.5 anys	0 €
ShareLaTeX	0 €	1	1 any	0 €
Subversion	0 €	1	1 any	0 €
Mendeley	0 €	1	1 any	0 €
TOTAL	**0 €**	-	-	**0 €**

Taula 3.2: Taula de descripció del cost del software.

3.1.3 Recursos humans

Aquest projecte el desenvoluparà només una persona. Com a conseqüència, aquesta persona haurà de fer el paper de totes les persones que estarien involucrades en un projecte d'aquestes característiques. S'ha suposat que en aquest treball hi participaria: **Cap de projecte**, amb un sou de 50€/hora; **Programador**, amb un sou de 35€/hora; i **Tester**, amb un sou de 30€/hora.

* Cedit per l'empresa PAL Robotics

COST RECURSOS HUMANS			
Tasca	Rol	Temps [h]	Preu
Definició del projecte	Cap de projecte	70	3500 €
	Programador	10	350 €
	Tester	0	0 €
	Total	**80**	**3850 €**
It0: Set Up inicial	Cap de projecte	5	250 €
	Programador	70	2450 €
	Tester	10	300 €
	Total	**85**	**3000 €**
It1: ASR	Cap de projecte	10.52	526 €
	Programador	73.70	2576 €
	Tester	15.78	473.40 €
	Total	**100**	**3575.40 €**
It2: Extractor semàntic	Cap de projecte	11.56	578 €
	Programador	81.10	2832.20 €
	Tester	17.34	526.20 €
	Total	**110**	**3930.40 €**
It3: Raonador	Cap de projecte	15.78	589 €
	Programador	110.54	3866.10 €
	Tester	23.67	710.10 €
	Total	**150**	**5165.20 €**
Redacció final de la memòria	Cap de projecte	14	700 €
	Programador	1	35 €
	Tester	0	0 €
	Total	**15**	**735 €**
Preparació de la presentació	Cap de projecte	23	1150 €
	Programadror	2	70 €
	Tester	0	0 €
	Total	**25**	**1220 €**
TOTAL		**565**	**21476 €**

Taula 3.3: Taula de descripció del cost de recursos humans

3.1.4 Despeses generals

A més del cost dels recursos analitzats, també hi ha un seguit de despeses generals que s'originaran durant la realització del treball. Les despeses analitzades en aquest apartat són: lloguer, aigua, electricitat, ADSL i transport. Per fer el càlcul del preu de cada despesa lligada al treball s'ha calculat el cost/hora de cada despesa i s'ha multiplicat pel nombre d'hores establertes a la planificació inicial.

DESPESES GENERALS			
Concepte	Preu	Temps	Total
Lloguer, aigua, electricitat	865 €/mes	565 hores	678.78 €
ADSL	45 €/mes	565 hores	35.31 €
Transport	35 €/mes	6 mesos	210 €
TOTAL ACUMULAT			**924.09 €**

Taula 3.4: Taula de descripció del cost de despeses generals.

3.1.5 Cost total

Finalment, s'ha sumat el cost dels quatre conjunts de recursos anteriorment analitzats i aplicat el I.V.A per estimar un preu total d'aquest treball final de grau. (S'aplica un I.V.A de tipologia general del 21%)

COST TOTAL PER CONCEPTE	
Concepte	Cost
Software	0 €
Hardware	70.55 €
Recursos Humans	21476 €
Despeses Generals	924.09 €
Subtotal	22470.64 €
I.V.A (21%)	4718.83 €
TOTAL	**27189.47 €**

Taula 3.5: Taula de descripció del cost total del treball.

3.2 Control de seguiment de costos

Establir un seguiment del cost del treball serà molt important per tal de verificar si el cost final és similar al del pressupost definit. Així doncs, al final de cada iteració s'analitzarà i calcularà el cost real en hores. El cost real del treball s'obtindrà de la suma de costs calculats al final de cada iteració. En el cas que durant el seguiment es detecti que no s'està complint el pressupost establert, s'hauran de reajustar els objectius i la planificació inicials per tal de compensar els costs en les següents tasques. Fos aquest el cas, sempre s'intentaria respectar el màxim les definicions establertes d'objectius i planificació.

3.3 Viabilitat econòmica

Aquest treball no té viabilitat en termes de beneficis per dos factors principals. El primer és que, tal com ja es va explicar, aquest treball es desenvolupa en el marc del projecte d'investigació REEM@IRI, una iniciativa sense ànim de lucre. El segon fet és que el resultat d'aquest treball no derivarà a un producte comercial i, per tant, no es recuperaran els costs que suposa el seu desenvolupament.

Tot i així, l'empresa PAL Robotics que ha cedit el robot REEM sí que podria treure un gran benefici econòmic d'aquest treball tot continuant amb el desenvolupament final de l'arquitectura. L'arquitectura que s'implementarà podria permetre millorar la intel·ligència dels seus robots i, en conseqüència, tenir una major possibilitat de venda dels seus robots. Per tant, aquest projecte seria econòmicament viable per a aquesta empresa.

En la Secció 4.2 s'analitza de forma explícita l'impacte econòmic del treball.

3.4 Pressupost final

En la Secció 2.3.5 es descrivia la planificació final del treball, la qual recollia tota una seria de modificacions respecte a la proposta inicial. Aquests canvis no han tingut afectació sobre els costs, ja que el treball ha estat dut a terme en el mateix temps que aquell estipulat inicialment (en quantitat d'hores dedicades).

Capítol 4

Sostenibilitat i compromís social

Un robot equipat amb l'arquitectura cognitiva que s'implementa en aquest treball ha de ser capaç d'entendre i completar tasques complexes ordenades en llenguatge natural. A més a més, l'escalabilitat d'aquesta arquitectura ha de permetre produir robots capaços d'ajudar en múltiples entorns i situacions noves. En conseqüència, aquest treball podria tenir una gran repercussió social, econòmica i ambiental.

4.1 Impacte social

Un dels entorns, sinó el principal, on un robot de servei podria ajudar és a les cases. Un robot dotat de la capacitat de recollir elements i desar-los, netejar superfícies i, eventualment, avisar de situacions d'emergència com ara un incendi, suposa una gran ajuda. La introducció en les cases d'un robot amb aquestes habilitats significaria una millora directa de la qualitat de vida de molta gent. Per exemple, seria molt beneficiós per la gent gran que viu sola, qui sovint té grans dificultats en la realització de les activitats de la vida diària a la llar.

Un altre entorn d'introducció d'aquests robots de servei és en fires i exposicions, aeroports, centres comercials, i espais públics en general. Així, els robots ajudarien i informarien els visitants/clients de les diferents activitats i espais disponibles. Resulta evident que la presència de robots en aquests entorns també suposaria un gran impacte social.

4.2 Impacte econòmic

Per analitzar l'impacte econòmic que podria suposar aquest treball, un bon cas d'estudi és el del robot aspirador *Roomba*[1] de l'empresa iRobot. Roomba ha tingut una gran acceptació social i s'han venut més de 8 milions d'unitats des de la seva sortida en el mercat fa 10 anys.

Roomba només oferia fins fa ben poc la possibilitat d'aspirar el terra. En canvi, l'arquitectura que s'implementa en aquest treball ha de permetre a un robot de servei ser capaç de realitzar tasques com servir begudes en un acte, rebre clients a una oficina, guiar persones per la ciutat, etc. Tenir robots dotats d'aquestes habilitats podria traduir-se a un gran increment en les compres de robots de servei per part de particulars, entitats i empreses. Per altra banda, la utilització d'aquests robots com a eina de màrqueting també podria suposar un augment de les vendes en altres sectors. Així doncs, es pot veure que aquest treball podria provocar un gran impacte econòmic.

4.3 Impacte ambiental

Com ja s'ha apuntat, aquest treball també podria tenir un impacte ambiental. Per exemple, aquests robots podrien ser útils en cas de desastres naturals ajudant a la recuperació de l'entorn. El cas més recent és el desastre a la central nuclear de Fukushima. A més a més, en aquestes situacions on s'han de realitzar tasques molt perilloses pels humans, es podrien enviar robots assignats als cossos d'emergència.

Tal com s'ha explicat en l'impacte econòmic, aquest treball podria significar un gran increment en la compra de robots de servei. Aquest fet provocaria un augment de la producció de robots i, com a conseqüència, una major quantitat de residus tecnològics. Per la qual cosa, aquest treball podria provocar de forma indirecta un impacte ambiental negatiu.

Tot i aquest possible impacte negatiu indirecte, l'arquitectura plantejada en aquest treball és "robot agnostic", és a dir, es pot introduir en qualsevol robot. Per tant, no depèn de cap hardware específic. Aquesta característica es tradueix en què es podrien produir nous robots amb components sobrants d'anteriors produccions o inclús amb components reciclats d'anteriors robots. Així doncs, aquesta propietat de l'arquitectura permetria disminuir l'empremta ecològica de possibles futures produccions en massa de robots de servei.

[1] www.irobot.com/global/es/roomba_range.aspx

4.4 Control de sostenibilitat

Tal com s'ha analitzat en els apartats anteriors, aquest treball podria provocar un gran impacte en l'àmbit social, econòmic i ambiental. És per això, que és important proposar un seguit de sistemes de control per tal d'intentar quantificar els efectes en sostenibilitat del treball.

Per tal de quantificar l'**impacte social** es podria realitzar una enquesta amb l'objectiu de mesurar l'acceptació en la realització de tasques d'aquests robots de servei en diferents entorns públics i privats. Per altra banda, també es podria fer un estudi introduint aquests robots en diferents cases i analitzar la millora de la qualitat vida dels seus habitants.

L'**impacte econòmic** que aquest treball podria provocar és principalment un augment de les vendes en el sector de la robòtica, però també un augment de les vendes en sectors que utilitzin aquests robots com a eina de màrqueting. Per tal de poder fer un primer estudi de l'impacte econòmic que es podria produir, es planteja el següent experiment. En dues fires similars s'estableixen dos expositors idèntics d'una mateixa marca. En un d'ells s'hi posa un comerciant i en l'altre un robot de servei equipat amb l'arquitectura implementada. En finalitzar el dia es fa un recompte del nombre de visites en cada un dels dos estancs així com una enquesta per tal d'analitzar la quantitat de visitants de cada fira que recorden la marca exposada.

Finalment, és difícil quantificar l'**impacte ambiental** indirecte del treball finalitzat. Tot i així, l'impacte directe sí que es podria quantificar. Per fer-ho, es proposa realitzar simulacions de possibles pertorbacions medi ambientals per tal de mesurar els efectes de la introducció d'aquests robots en l'ajuda a la recuperació del medi afectat.

Capítol 5

Descripció de l'arquitectura cognitiva

L'arquitectura implementada en aquest treball es compon d'una interfície i de tres mòduls principals connectats entre si (veure Figura 5.1). En primer lloc, una ordre de veu és rebuda pel robot i es tradueix a text mitjançant l'ús d'un sistema de reconeixement de veu automàtic en el mòdul *ASR*. A continuació, el mòdul *Extractor semàntic* divideix el text rebut en estructures gramaticals i es genera un goal o objectiu a partir d'elles. Posteriorment, el goal és compilat en el mòdul *Raonador* i s'envia a l'arquitectura cognitiva Soar. Aquest mòdul va seleccionant l'habilitat que el robot ha d'executar en cada instant de temps (veure Taula 1.1). La *Interfície* rep l'habilitat seleccionada i l'executa mitjançant els nodes d'acció del sistema operatiu del robot. Finalment, la interfície captura el resultat de l'execució de l'habilitat i li envia al mòdul *Raonador* per tal que pugui actualitzar el nou estat del seu entorn.

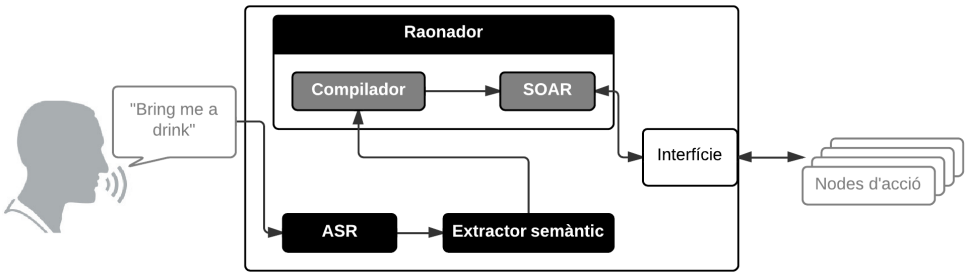


Figura 5.1: Diagrama de l'arquitectura

Per facilitar la comprensió de la interacció entre els diferents mòduls així com el comportament complet de l'arquitectura, s'exemplificarà tot el procés suposant que es diu al robot l'ordre "bring me a drink". Es pot veure l'execució d'aquest exemple en vídeo en el link: `http://youtu.be/piCqb2O7rtU`.

5.1 Mòdul ASR

Per tal de permetre la comunicació mitjançant la veu, el sistema incorpora un reconeixedor automàtic de la parla (ASR). Aquest mòdul s'encarrega de processar els senyals de veu i retornar-los com una cadena text per una posterior anàlisi semàntica. Per tant, aquest mòdul proporciona una *interacció natural humà-robot* (HRI).

La Figura 6.2 mostra la interfície gràfica del mòdul *ASR*.

Exemple: Tal com s'ha indicat, la comanda verbal al robot és "bring me a drink". Aquest mòdul rep la senyal sonora de veu i retorna la cadena de text <bring me a drink>.

5.2 Mòdul Extractor semàntic

El mòdul *Extractor semàntic* és l'encarregat de processar les oracions imperatives rebudes en forma de text des del mòdul *ASR* i així identificar el rol gramatical de cada paraula en l'oració (verb, subjecte, etc).

Cada ordre pot està formada per una o més oracions. Cada oració representa una subordre. Aquest mòdul extreu les subordres contingudes en el text a partir dels connectors gramaticals. Els connectors possibles són: la conjunció ("and"), la partícula de transició ("then") o signes de puntuació. Cal tenir en compte però, que atès que l'oració analitzada prové del mòdul *ASR*, tots els signes de puntuació han estat prèviament omesos. Tot i així, també s'ha implementat la separació de subordres per signes de puntuació, ja que futurs usuaris de l'arquitectura podrien preferir entrar l'ordre directament en format text en comptes de per veu.

L'arquitectura implementada permet dos tipus d'oracions (subordres):

- **Categoria I** L'oració representa una habilitat específica del robot. Per exemple, "go to the kitchen then search a person and introduce yourself". Les accions permeses en una subordre són les definides en la Taula 1.1.

- **Categoria II** L'oració no és prou específica per falta d'informació necessària o per contenir categories de paraules en lloc d'objectes específics. Per exemple, "bring a coke to someone" o "bring me a drink". El primer exemple no inclou informació sobre qui és el receptor de la beguda. En el segon exemple no s'identifica quina beguda s'està demanant. A més a més, a diferència de les oracions de *Categoria I*, la subordre no especifica quina habilitat s'ha d'executar per a completar la tasca demanada.

S'ha suposat que les ordres sempre s'expressen amb frases imperatives, ja que el que s'espera és que el robot realitzi una o més accions. Les accions a ser realitzades pel robot sempre es detecten a partir dels verbs perquè en frases imperatives els verbs han de ser accions. D'altra banda, també s'ha aplicat l'hipòtesis simplificadora que qualsevol subordre implicarà al robot en la realització d'una acció sobre un únic objecte determinat ("grasp a coke"), una única localització específica ("navigate to the kitchen table") i/o una única persona concreta ("bring me a drink").

Així doncs, cada subordre està composta per la següent informació principal:

- Quina acció s'ha de dur a terme;
- Quina localització és rellevant per a l'acció determinada (si n'hi ha).
- Quin objecte és rellevant per a l'acció determinada (si n'hi ha).
- Quina persona és rellevant per a l'acció determinada (si n'hi ha).

Finalment, aquest mòdul envia la informació de cada subordre formada per aquests quatre camps al mòdul *Raonador*.

Exemple: Aquest mòdul rep l'ordre "bring me a drink" i la 'parseja' (analitza sintàcticament). Envia al mòdul *Raonador*: **acció**: bring; **localització**: null; **objecte**: drink; **persona**: persona davant el robot.

5.3 Mòdul Raonador

Aquest mòdul està dividit en dos submòduls: mòdul *Compilador* i mòdul *Soar*. Aquests, són descrits en els següents apartats.

5.3.1 Compilador

El submòdul *Compilador* rep la informació enviada pel mòdul *Extractor semàntic* i la transforma en goals comprensibles pel submòdul Soar.

Així doncs, primerament aquest submòdul rep la informació emesa pel mòdul *Extractor semàntic*. Tal com s'ha explicat anteriorment, aquesta informació és un conjunt d'estructures (una per cada subordre que forma la comanda) on cada una conté la descripció de l'acció ha ser realitzada i informació de la localització, objecte i/o persona involucrats en l'acció.

Posteriorment, en el cas que una o més d'aquestes estructures provingués d'una oració de *Categoria II*, és possible qui hi hagi informació necessària que hi manca. Per tant, aquest mòdul identifica possible informació mancant necessària, i la sol·licita. Per exemple, en l'orde "point at a drink", el mòdul identificarà que no s'ha especificat quina beguda en concret s'ha d'apuntar i la demanarà.

També és possible que falti informació sobre la localització de l'objecte que intervé en la tasca. Primerament, aquesta informació es busca en una ontologia del robot on es llisten les localitzacions dels objectes de l'entorn. En el cas que l'objecte buscat no es trobi en l'ontologia, se li demana la seva localització a la persona. Cal destacar que aquesta informació és necessària perquè actualment el robot REEM no té cap algoritme implementat de cerca d'objectes.

Finalment, un cop obtinguda tota la informació necessària, els objectius es compilen i s'envien al mòdul *Soar*.

Exemple: Aquest mòdul rep el següent missatge enviat pel mòdul *Extractor semàntic*: **acció**: bring; **localització**: null; **objecte**: drink; **persona**: persona davant el robot. Posteriorment, aquest mòdul demana informació sobre quina beguda es desitja; la persona respon 'coke'. Finalment, un cop processada la resposta, compila el goal com un estat del món on l'objecte *coke* està entregat a la persona que ha donat l'ordre.

5.3.2 Soar

El submòdul *Soar* és l'encarregat de determinar quines habilitats han de ser executades per tal d'assolir el goal compilat. Un bucle dins de Soar selecciona l'habilitat que ha d'executar el robot en cada moment per apropar-se un pas més al goal. Cada vegada que se selecciona una habilitat, una petició d'execució d'aquella habilitat és enviada a la *Interfície*. Un cop finalitzada l'execució de l'habilitat, Soar rep de la *Interfície* si l'habilitat s'ha executat correctament o no. A partir d'aquesta informació, Soar actualitza l'estat del món i selecciona una nova habilitat. Aquest procés es va repetint fins que s'assoleix el goal.

Exemple: En la primera iteració aquest mòdul selecciona l'habilitat: *go to kitchen table*. Un cop executada la primera habilitat, la *Interfície* retorna que l'exe-

cució ha estat satisfactòria i aquest mòdul actualitza l'estat del món definint que el robot es troba davant de la taula de la cuina. En la segona iteració selecciona: *search coke*. Un cop rep el missatge que l'habilitat s'ha executat correctament, actualitza l'estat del món definint que la 'coke' sí es troba sobre la taula. Seguidament selecciona l'acció *grasp coke*. Un cop rep el missatge que l'execució ha estat satisfactòria, actualitza l'estat del món definint que ara el robot ja té la "coke". Un cop agafada la beguda, selecciona l'acció *go to the person*, rep el missatge d'execució satisfactòria i actualitza el món. Finalment, selecciona l'habilitat *deliver coke*. Executada correctament l'habilitat, l'estat actual del món (s'ha entregat la beguda a la persona) coincideix amb el goal ("bring me a coke") i, per tant, finalitza l'execució de l'ordre i espera fins a rebre una nova per tornar a començar tot el procés.

5.4 Interfície Soar – Nodes d'acció

Aquesta interfície és l'encarregada de gestionar la comunicació entre el submòdul *Soar* i els *Nodes d'acció*. Aquesta interfície és necessària perquè no existeix una implementació de Soar per ROS. Així doncs, aquesta interfície és l'encarregada de capturar les habilitats proposades per Soar a ser executades i cridar al *Node d'acció* que implementa l'habilitat. A més a més, també s'encarrega de capturar el feedback de l'execució de l'habilitat i enviar-la a Soar per tal de què pugui actualitzar l'estat actual del món.

5.5 Nodes d'acció

Els *Nodes d'acció* són les peces modulars de programari que implementen les habilitats del robot (veure Taula 1.1). Per executar una habilitat del robot només s'ha de cridar al node que implementa aquella habilitat passant-li els paràmetres corresponents. Un cop realitzada l'execució, aquest retorna si l'execució ha estat satisfactòria o no.

Capítol 6

Implementació de l'arquitectura

En aquest capítol es descriu la implementació de l'arquitectura. La figura 6.1 mostra el diagrama de la implementació.

6.1 Representació del món

La representació del món està dividida en els coneixements del robot del seu entorn i del tractament d'aquesta informació dintre del submòdul *Soar*.

Hi ha cinc tipus d'informació que el robot codifica del món per tal de poder operar correctament.

1. Un mapa de l'entorn en què el robot ha de treballar. Aquest mapa es fa servir per navegar pels llocs. Per tant, el mapa també conté la informació sobre els llocs en els quals pot realitzar accions. Exemple de localitzacions inclouen sales específiques com *kitchen* o *dining room*. També conté objectes

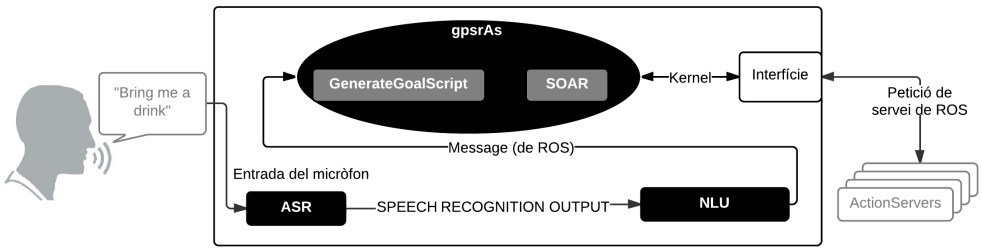


Figura 6.1: Diagrama de la implementació de l'arquitectura

fixos específics amb els quals pot interactuar com per exemple *kitchen table*, *trash bin* o *entry door*.

2. Una ontologia que conté totes les accions, noms d'objectes, noms de persones i noms de llocs que el robot pot entendre a través del mòdul *Extractor semàntic*. Aquesta ontologia inclou quines accions són aplicables a quins elements (llocs, persones, objectes), la ubicació per defecte d'alguns objectes i la categoria d'alguns objectes i llocs (per exemple, un objecte podria ser classificat com begudes o aliments; mentre que una ubicació podria ser un seient, una habitació o una taula).

3. Una base de dades amb els models dels objectes que el robot és capaç de reconèixer. Es tracta d'una base de dades de models 2D/3D dels objectes.

4. Una base de dades de les cares que el robot és capaç de reconèixer.

A part d'aquesta informació, l'arquitectura també necessita una representació de l'estat actual del món pròpia pel submòdulSoar. Aquest estat es defineix com una base de dades simple, dins de l'arquitectura cognitiva, amb el coneixement actual sobre l'estat del món, incloent-hi l'estat del robot, objectes, localitzacions i persones. La informació que conté és només la necessària per establir si un goal ha estat completat.

- robot
  - identificador del robot
  - localització actual
  - localització apuntada
  - s'ha presentat a si mateix?
  - objecte actualment agafat
- objecte
  - identificador de l'objecte
  - localització actual
  - ha estat apuntat?
  - ha estat localitzat?
  - ha d'agafar-se i ha estat agafat?
- localització
  - ha estat apuntada?
- persona

- identificador de la persona
- localització actual
- ha estat localitzada?
- ha estat reconeguda?
- està en la localització actual del robot?
- se li a preguntat el nom?
- objecte entregat a la persona

6.2 Mòdul ASR

Aquest mòdul està implementat en *c++* i utilitza la llibreria *LCM*[1] per la comunicació. Aquest mòdul llegeix l'entrada del micròfon seleccionat en la interfície en pitjar el botó *press and hold to record*. L'arxiu d'àudio que conté la comanda és introduït en la llibreria *Google speech recognition* (veure Secció 1.4.1) i aquesta retorna el *string* (cadena de text) corresponent a la traducció d'àudio a text. El funcionament d'aquesta llibreria no és públic, així doncs, la part de traducció de l'ordre de veu a una oració escrita és una caixa negre. Cal destacar que el *string* retornat no conté signes de puntuació, ja que són un element artificial de l'escriptura. Finalment, publica aquest *string* en un *channel* de *LCM* anomenat *SPEECH_RECOGNITION_OUTPUT* en el qual el mòdul *Extractor semàntic* està subscrit.

Exemple: Aquest node llegeix del micròfon l'ordre "bring me a drink". Genera el fitxer d'àudio i l'introdueix a la llibreria *Google speech recognition* la qual retorna el *string* <bring me a drink>;. El mòdul publica aquest *string* en el *channel* *SPEECH_RECOGNITION_OUTPUT*.

6.3 Mòdul Extractor semàntic

Aquest mòdul anomenat *NLU* està implementat en *c++* i utilitza la llibreria *LCM* per la comunicació. Aquest mòdul va rebent noves ordres verbalitzades al robot a través *channel SPEECH_RECOGNITION_OUTPUT* on el node *ASR* hi publica.

Aquest node utilitza *h2sl* (veure Secció 1.4.2) per analitzar sintàcticament les ordres. *h2sl* està basat en grafs distribuïts de correspondència [10] expressat com

[1] Lightweight Communications and Marshalling

una combinació ponderada de característiques. Per definir els pesos de les característiques del model s'ha hagut d'entrenar a base d'exemples. Aquests exemples han de contenir el món del model i una orde en llenguatge natural parsejada. En la Figura 6.3 es mostra un exemple d'un fitxer d'entrenament. Cada fitxer d'entrenament es converteix a un *annotated parse tree* tal com es pot veure en la Figura 6.3. Per entrenar el NLU s'ha utilitzat un total de 33 exemples. Un cop entrenat, *h2sl* genera un model i el guarda. Aquest model es carrega en iniciar l'arquitectura.

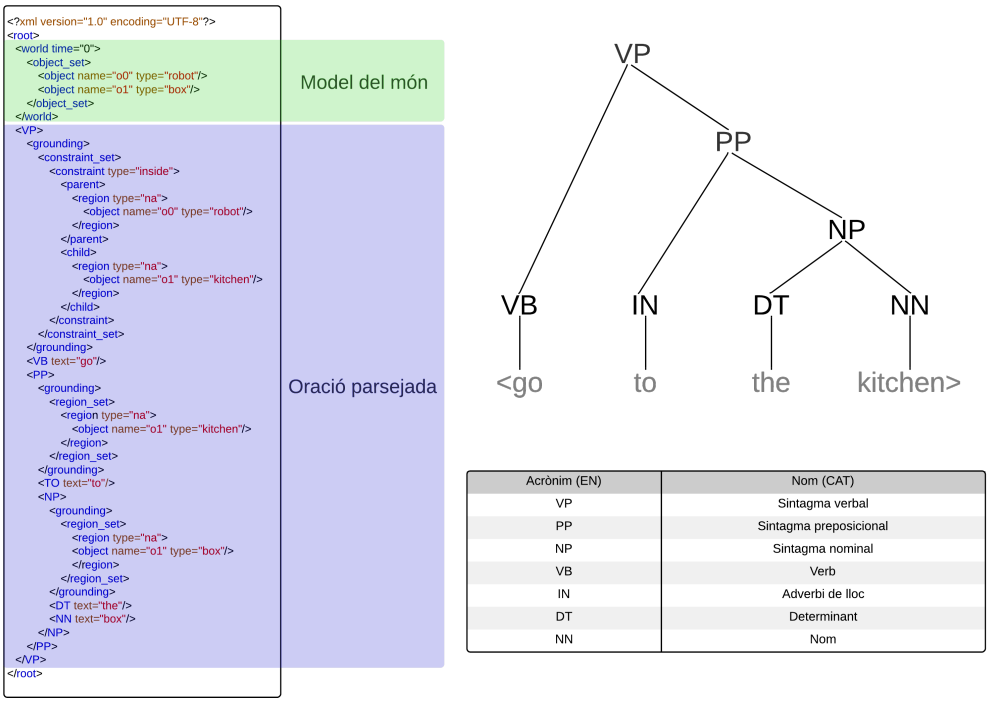
Cada nova ordre que arriba al mòdul és parsejada amb *h2sl* seguint el complex algoritme probabilístic descrit en detall a [16]. Un cop la comanda està parsejada (analitzada sintàcticament), l'obtenció de l'acció és immediata perquè és el nucli del sintagma verbal. El següent pas és l'obtenció dels seus complements (objecte, localització i/o persona). Per tal d'aconseguir-ho, es realitza una combinació de dos mètodes:

1. Una ontologia permet identificar quines paraules de l'oració són objectes, persones o llocs. Aquesta ontologia forma part del coneixement bàsic del robot i és gestionada pel mateix robot. L'arquitectura implementada hi té accés.

2. Dependències entre les paraules en l'oració. *h2sl* en estar basat en grafs distribuïts de correspondència expressats com una combinació ponderada de característiques, permet identificar quines parts de l'oració estan connectades entre si i, en aquest cas, identificar quins connectors tenen. Per tant, permet detectar quins substantius dintre del sintagma verbal actuen com a objectes directes i, inclús, objectes indirectes i adverbis de lloc.

Finalment, en aquest punt l'ordre ja ha estat parsejada en subordres i per cada subordre s'han identificat els camps definits anteriorment (acció, localització, objecte i/o persona), es genera un *message* de ROS amb el següent format:

Figura 6.2: Interfície gràfica ASR



Acrònim (EN)	Nom (CAT)
VP	Sintagma verbal
PP	Sintagma preposicional
NP	Sintagma nominal
VB	Verb
IN	Adverbi de lloc
DT	Determinant
NN	Nom

Figura 6.3: Exemple fitxer d'entrenament, analisi sintàctic del fitxer i taula de traducció dels acrònims

```
\textit{string} action
\textit{string} localization
\textit{string} object
\textit{string} person
```

Aquest *message* es capturat pel node de ROS *gpsrAs* (submòdul *Compilador*).

Exemple: Aquest mòdul rep l'ordre <bring me a drink> i l'introdueix a la llibreria *h2sl*. Aquesta llibreria realitza un anàlisi sintàctic de l'ordre. Detecta que *bring* com el nucli del sintagma verbal; *drink* com l'objecte directa i; *me* com l'objecte indirecta. Finalment, genera un *message* amb els camps: **action**: bring; **localization**: null; **object**: drink; **person**: referee.

6.4 Mòdul Raonador

Aquest mòdul és un *node* de ROS anomenat *gpsrAs*. Aquest *node* recull el *message* del mòdul *Extractor semàntic*. Quan aquest *node* està actiu, va capturant la

sortida generada pel mòdul *Extractor semàntic*(goals). Cada nou goal que captura l'inserta en una cua de goals pendents; quan acaba un goal, desempila de la cua el següent goal ha ser realitzat amb ordre FIFO. Per cada goal executa el script *GenerateGoalScript* (submòdul Compilador) el qual compila el goal. Un cop el goal està compilat, s'inicia l'execució de submòdul Soar.

6.4.1 Compilador

Aquest submòdul forma part del *node gpsrAs*. Cada cop que aquest *node* rep un nou goal executa aquest submòdul per tal que el compili. Aquest submòdul està implementat en Python.

Primerament, quan aquest *node* rep un nou missatge fa la comprovació de si hi ha informació mancant necessària. La informació mancant es pot donar per dos casos:

1. En comptes d'un objecte concret la persona ha dit una categoria.
2. Per a l'acció demanada, és necessari algun camp (objecte, localitat i/o persona) que no s'ha donat.

Cas 1 Aquest cas és detectat mirant a l'ontologia si l'objecta donat és un objecte concret o una categoria. En el cas que es detecti que és una categoria, s'identifica la informació del camp *objecte* com a necessària i mancant. En conseqüència, generà una pregunta on s'informa la persona de tots els objectes de la categoria donada i se li pregunta que a quin es refaria. Per realitzar aquesta pregunta, s'envia una petició al *node* del robot encarregat de la parla del robot amb la pregunta en format *string*. Posteriorment, el *node* encarregat de la parla retorna un *string* amb la resposta de la persona. Un cop ha rebut la resposta, busca si és un objecte vàlid en l'ontologia. En el cas que sigui un objecte vàlid es guarda. En cas contrari, es torna a fer la pregunta.

Cas 2 Aquest cas es detecta amb unes plantilles predefinides on per cada possible goal s'identifica quina informació és necessària. Un cop detectada la informació mancant, es demana. En els dos casos, un cop la persona a respòs, es realitza una comprovació de la resposta buscant l'objecte, localització o persona en l'ontologia. En cas que no existeixin s'informa la persona i se li torna a preguntar.

Un cop es té tota la informació mancant, es realitza la compilació de l'ordre. Per fer-ho, aquest *node* té prèviament definides un seguit de plantilles que li permeten fer la traducció del missatge rebut a goals de Soar. Hi ha dos tipus de plantilles: plantilles generals que s'aplicant a qualsevol ordre; i plantilles específiques que s'apliquen a ordres concretes. Així doncs, un cop rebut un missatge, s'agafa l'acció

ha ser realitzada i es busca les plantilles necessàries per compilar l'ordre. Un cop obtingudes, introdueix la informació respectiva de l'objecte, localitat i/o persones involucrades en l'ordre. Finalment, es publica la sortida de les plantilles (goal compilat) en un arxiu anomenat "initialize-gp.Soarï es guarda en l'arquitectura Soar.

Exemple: Seguint l'exemple definit anteriorment, en l'ordre "bring me a drink", el *node gpsrAs* captura un nou goal amb els camps: **action**: bring; **localization**: null; **object**: drink; **person**: referee. Un cop rebut el goal, comprova en la seva ontologia si "drink"és un objecte concret. En veure que és una categoria d'objectes, i no un objecte concret, identifica la informació del camp *object* com a necessària i mancant. En conseqüència, envia una petició al *node* encarregat de la parla del robot amb la pregunta on informa la persona de totes les begudes de la categoria "drinkï li pregunta que quina desitja. El *node* encarregat de la parla del robot retorna la resposta, *string* <coke>. A continuació, comprova en l'ontologia que 'coke' sí que és un objecte concret i es guarda. També busca la localització de la 'coke' en l'ontologia. En aquest punt, ja té tota la informació necessària per compilar l'ordre. Per tant, busca les plantilles necessàries per l'acció "bring" i genera el goal en el fitxer *initialize-gp.Soar* com un estat del món on l'objecte *coke* està entregat a la persona que ha donat l'ordre. A continuació es mostra el goal compilat:

```
sp {gp*propose*initialize-gp
   (state <s> ^superstate nil
              - ^name)
-->
   (<s> ^operator <op> +)
   (<op> ^name initialize-gp)
}

sp {gp*apply*initialize-gp
   (state <s> ^operator.name initialize-gp)
-->
   (<s> ^name gp
        ^location <l0> <l1> <l2> <l3> <l4> <l5> <l6> <l7> <l8> <l9> <l10> <l11> <l12> <l13> <l14> <
            l15> <l16> <l17> <l18> <l19> <l20> <l21> <l22> <l23> <l24> <l25> <l26> <l27> <l28>
        ^robot <r>
        ^object <obj1>
        ^person <pers1>
        ^desired <d>)
   (<l0> ^id 0 ^pointed-at no) (<l1> ^id 1 ^pointed-at no) (<l2> ^id 2 ^pointed-at no)
   (<l3> ^id 3 ^pointed-at no) (<l4> ^id 4 ^pointed-at no) (<l5> ^id 5 ^pointed-at no)
   (<l6> ^id 6 ^pointed-at no) (<l7> ^id 7 ^pointed-at no) (<l8> ^id 8 ^pointed-at no)
   (<l9> ^id 9 ^pointed-at no) (<l10> ^id 10 ^pointed-at no) (<l11> ^id 11 ^pointed-at no)
   (<l12> ^id 12 ^pointed-at no) (<l13> ^id 13 ^pointed-at no) (<l14> ^id 14 ^pointed-at no)
   (<l15> ^id 15 ^pointed-at no) (<l16> ^id 16 ^pointed-at no) (<l17> ^id 17 ^pointed-at no)
   (<l18> ^id 18 ^pointed-at no) (<l19> ^id 19 ^pointed-at no) (<l20> ^id 20 ^pointed-at no)
   (<l21> ^id 21 ^pointed-at no) (<l22> ^id 22 ^pointed-at no) (<l23> ^id 23 ^pointed-at no)
   (<l24> ^id 24 ^pointed-at no) (<l25> ^id 25 ^pointed-at no) (<l26> ^id 26 ^pointed-at no)
   (<l27> ^id 27 ^pointed-at no) (<l28> ^id 28 ^pointed-at no)
    (<r> ^pointedAtLoc -1
       ^introduced no
```

```
      ^locId 27
      ^id 1
      ^obj1Id -1)
   (<obj1> ^delivered no
      ^pointed no
      ^locId 6
      ^toBeGrasped no
      ^grasped no
      ^found no
      ^id 6)
   (<pers1> ^followed no
      ^memorized no
      ^locId 27
      ^obj1Id -1
      ^recognized no
      ^askedName no
      ^near no
      ^found no
      ^id 5)
   (<d> ^name bring-to
       ^person <pp>
       ^object <ii>
       ^robot <rr>)
   (<pp> ^obj1Id 6
         ^id 5)
   (<ii> ^delivered yes
         ^id 6)
   (<rr> ^id 1)
}
```

6.4.2 Soar

Aquest submòdul forma part del *node gpsrAs*. Cada cop que aquest *node* a compilat un nou goal executa aquest submòdul per tal de planificar i completar el goal. Aquest submòdul està implementat en el llenguatge propi de Soar.

Aquest submòdul executa l'arquitectura Soar. En l'arquitectura Soar s'han definit prèviament totes les habilitats que el robot pot realitzar (veure Taula 1.1). Aquestes es codifiquen com una llista d'operadors. Per tant, per a cada habilitat possible es defineix:

- Una norma que proposa l'operador. Aquesta conté el nom i els atributs corresponent.
- Una regla que envia la sortida que el robot executi aquesta habilitat si s'accepta el seu operador. Aquesta sortida conte el nom de l'habilitat juntament amb la informació complementaria. Per exemple, si l'habilitat és 'grasp', la sortida contindrà 'grasp' més l'objecte a ser agafat.
- Una o diverses regles que estableixen com afecta l'execució de l'habilitat definida en el món.

En la Figura 6.4 es mostra un exemple de definició d'una l'habilitat. L'exemple correspon a l'habilitat *grasp*.

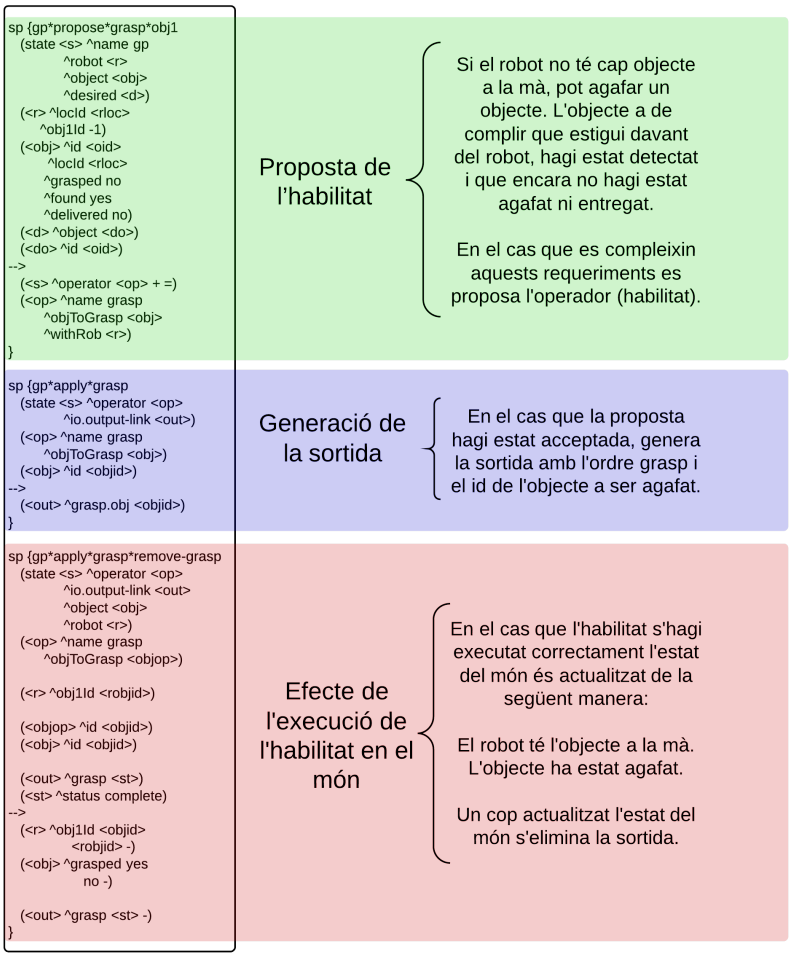


Figura 6.4: Definició de l'habilitat *grasp*

L'estructura de Soar es basa en un bucle on en cada iteració es decideix la següent habilitat ha ser executada. L'Algorisme 1 mostra el pseudocodi del bucle de Soar.

Primerament, es comproven totes les normes que proposen un operador. Totes les propostes són tractades alhora i es comparen en termes de preferències. Es selecciona la proposta que té una major prioritat, i s'executarà l'operador corresponent a la proposta. És important destacar que totes les regles proposades són tractades com si fossin disparades al mateix instant de temps, en paral·lel.

Un cop seleccionada l'habilitat que el robot ha d'executar (la corresponent a la proposta seleccionada), es genera com a sortida l'ordre que el robot ha d'executar aquella habilitat. Aquesta sortida conté l'habilitat i l'objecte, lloc i/o persona

Algorisme 1 Bucle Soar

```
1: goal ← not accomplished
2: while goal not accomplished do
3:     proposals ← TestProposals()
4:     selectedProposal ← MaxPriority(proposals)
5:     GenerateOutput(selectedProposal)
6:     while notFeedbackRecived do nothing
7:     end while
8:     if feedback = success then
9:         UpdateWorld(selectedProposal)
10:        goal ← TestGoalAccomplished()
11:    else
12:        Failed(selectedProposal)
13:    end if
14: end while
```

involucrada en l'habilitat. Soar requereix un estat del món actualitzat per tal de decidir la següent habilitat ha ser executada. Així doncs, l'estat del món s'actualitza després de cada execució d'una habilitat.

El món podria ser canviat pel robot o per altres agents existents (persones, altres robots, etc). Els canvis realitzats en el món pel robot es reflecteixen directament en el resultat de l'execució de les seves habilitats. Per aplicar aquests canvis al món, un cop l'ordre ha sigut enviada, s'espera una resposta de la *Interface* que contingui el resultat de l'execució de l'habilitat. En cas que l'habilitat s'hagi executat correctament, s'apliquen les regles que modifiquen l'estat del món. En cas que hagi fallat, es torna a proposar aquella habilitat. Es torna a proposar la mateixa habilitat perquè el robot on s'ha realitzat l'experimentació (REEM) no és capaç de retornar el perquè ha fallat l'execució d'una habilitat i, per tant, l'execució pot haver fallat per una gran quantitat de motius diferents. Així doncs, s'ha definit que quan una habilitat falla es torna a proposar aquesta mateixa habilitat fins a un total de tres cops. Si els tres cops falla, s'intenta proposar una nova habilitat disminuït la prioritat de l'habilitat que s'ha intentant executar tres cops.

Per altra banda, quan els canvis en el món són causats per altres agents que no són el robot són molt difícils de detectar i tractar. Per detectar-los seria necessari que el robot estigués equipat amb un software que li permetés detectar canvis en l'entorn, però aquest treball no entrarà en aquest tema per la gran complexitat d'aquest software. Així doncs, en desconèixer els possibles canvis en el món fets per altres agents, el robot podria fallar l'execució d'una habilitat perquè no coincideix el món real amb el representat en l'arquitectura. Aquesta situació provocaria l'execució d'una altra habilitat que tractaria de resoldre l'impasse[2]. Per exemple, en el cas en

[2]Cas en què no es pogués decidir la millor acció a realitzar amb el coneixement disponible perquè no hi ha cap proposta d'operador amb una major prioritat que les altres

què el robot s'envia a una posició on es creu que es troba un objecte amb el qual es vol interactuar però en arribar a aquella posició l'objecte no hi és. Per resoldre aquest impasse, es dispararà l'habilitat de cercar d'objectes per tal d'esbrinar la nova posició de l'objecte.

També pot passar que no hi hagi un pla per aconseguir el goal. Soar té implementats diversos mecanismes per a resoldre aquestes situacions. En aquest treball s'han activat els dos primers mecanismes. L'ús de *Reinforcement learning* es proposa com a mecanisme ha ser analitzat en un treball futur.

1. **Sub-goal capacity:** [18] Permet que el robot trobi una manera de sortir d'un impasse amb les habilitats disponibles per tal d'intentar apropar-se al goal.

2. **Chunking ability:** [18, 11, 34] Permet la producció de noves regles que ajuden el robot a adaptar-se a noves situacions i poder executar nous objectius mai afrontats abans.

3. **Reinforcement learning:** [27] Juntament amb les dues característiques anteriors, ajuda el robot a aprendre a realitzar en menys passos objectius executats anteriorment.

Finalment, es compara l'estat del món actual amb l'estat desitjat (goal). En el cas que siguin igual, s'ha assolit el goal. En el cas contrari, s'inicia una nova iteració.

En aquest treball s'han definit un total de deu habilitats diferents descrites a la Taula 1.1. En cada iteració de decisió d'una habilitat es proposen un total de 77 normes.

Exemple: Aquest exemple correspon a l'estat en què el robot ja està davant de la beguda i la detectada. Primerament, es generen totes les propostes. L'habilitat *grasp* és una de les proposades, ja que compleix totes les condicions: el robot no té cap objecte a la mà i està davant de l'objecte ha ser agafat prèviament detectat. També es proposen altres habilitats com desplaçar-se a una altra localització perquè també compleixen els requisits necessaris. Tot i així, *grasp* té una prioritat major i en conseqüència s'executa el seu operador *apply*. Aquest operador genera una sortida amb l'ordre *grasp* juntament amb l'identificador de l'objecte ha ser agafat. Aquesta sortida la captura la *Interface*. Un cop la *Interface* ha executat l'acció, retorna el seu resultat. Soar rep el resultat i en cas que sigui *success* s'executa l'actualització del món on es defineix que ara el robot té la beguda i l'objecte ja ha estat agafat. En el cas que fos *aborted* es tornaria a proposar l'habilitat fins a un màxim de tres cops. Finalment, es comprova que l'estat actual del món no és el mateix que el desitjat i, per tant, s'inicia una nova iteració del bucle de Soar.

6.5 Interfície Soar – Nodes d'acció

Aquesta interfície està implementada en Python i gestiona la comunicació entre el submòdul *Soar* i els *Nodes d'acció*. Per poder comunicar-se amb Soar aquesta interfície genera un kernel amb Soar com a agent. L'Algorisme 2 mostra el seu pseudocodi.

En cada iteració del bucle es capturen totes les sortides generades pel submòdul *Soar* i es processen. Per una banda el *name* de la sortida és l'habilitat a ser executada, i el *parameter* és la informació associada (objecte, localització o persona). Un cop processada la sortida, es crida el *Node d'acció* que implementa l'execució de l'habilitat amb els paràmetres corresponents. Finalment, en el cas que sigui *success* es retorna que s'ha completat l'ordre i, per contra, en el cas que falli, es retorna que s'ha produït un error.

6.6 Nodes d'acció

Els *Nodes d'acció* són ActionServers. Els ActionServers són nodes de ROS que permeten a nodes(clients) fer peticions d'execució a altres nodes(servers) i rebre un missatge amb el feedback de l'execució de la petició. Cada vegada que el sistema Soar proposa una habilitat per a ser executada, la *Interfície* la rep i genera una petició al ActionServer a càrrec de l'habilitat proposada. En finalitzar l'execució de l'habilitat, el server retorna un feedback sobre el resultat de l'execució (*success* o *aborted*). Aquesta retroalimentació és capturada per la *Interfície*.

Algorisme 2 Pseudocodi Interfície

```
1: kernel ← CreateKernel()
2: agent ← CreateAgent(kernel)
3: LoadProductions(agent,SoarPath)
4: goal ← not accomplished
5: while goal not accomplished do
6:     agent.RunTilOutput()
7:     agent.Commands()                          ▷ Captura sortida de Soar
8:     i ← agent.GetNumberOfCommands()
9:     while i ≠ 0 do
10:        command ← command.GetCommand(i)
11:        feedback ← Execute(command)      ▷ Crida Nodes d'acció de l'habilitat
12:        agent.AddStatus(feedback)              ▷ Retorna el feedback a Soar
13:        i ← i-1
14:    end while
15: end while
```

Capítol 7

Experimentació i resultats

7.1 Execució completa

Per tal de comprovar el correcte funcionament de l'arquitectura s'ha introduït el robot en un entorn domèstic i se li han dit un conjunt d'ordres de Categoria I i II. A continuació es mostren algunes d'aquestes ordres:

Categoria I. Oració completa sense informació mancant.

- "*Go to the kitchen, find a Coke and grasp it*"
  Seqüència d'accions realitzades pel robot: *understand command, go to kitchen, search for coke, grasp coke*
- "*Go to reception, find a person and introduce yourself*"
  Seqüència d'accions realitzades pel robot: *understand command, go to reception, search person, go to person, introduce yourself*
- "*Find the closest person, introduce yourself and follow the person in front of you*"
  Seqüència d'accions realitzades pel robot: *understand command, search person, go to person, introduce yourself, follow person*

Categoria I amb errors Oració completa sense informació mancant amb errors.

- "*Go to the sun*"
  Seqüència d'accions realitzades pel robot: *inform: location sun not defined*
- *"Go to reception, find a person and introduce yourself"però no hi ha cap persona*
  Seqüència d'accions realitzades pel robot: *understand command, search*

person, search person, search person, inform: unable to find a person

Categoria II Oració amb informació mancant necessària.

- "*Carry a snack to a table*"
  Seqüència d'accions realitzades pel robot: *understand command, ask questions, acknowledge all information, go to location, search snack, grasp snack, go to table, deliver snack*

- "*Bring me a drink*"(Figura 7.1)
  Seqüència d'accions realitzades pel robot: *understand command, ask questions about which drink, acknowledge all information, go to location, search energy drink, grasp energy drink, go to origin, deliver energy drink*

Categoria II amb errors Oració amb informació mancant necessària amb errors.

- "*Bring me a cow*"
  Seqüència d'accions realitzades pel robot: *inform: object cow not defined*

- *"Bring me a drink"però no hi ha cap beguda*
  Seqüència d'accions realitzades pel robot: *understand command, ask questions about which drink, acknowledge all information, go to location, search energy drink, search energy drink, search energy drink, inform: unable to find the energy drink*

Dels resultats obtinguts durant la fase d'experimentació es conclou el següent. No es pot afirmar que la seqüència d'accions sigui l'òptima dagut de la naturalesa cognitiva de Soar. Tot i això, sí es pot garantir la finalització de la tasca ordenada, ja que l'arquitectura continuarà proporcionant passos fins que aconsegueixi completar-la.

Es pot veure l'execució de l'exemple *Bring me a drink* que s'ha anat detallant al llarg del treball en el link: `http://youtu.be/piCqb2O7rtU`

7.2 Incloure noves habilitats

Actualment els robots de servei evolucionen a gran velocitat. Les millores de software i hardware són constants gràcies a la recerca del sector i els robots de servei adquireixen noves habilitats molt freqüentment. Per aquest motiu és bàsic que l'arquitectura implementada sigui suficientment canviable com per poder afegir noves habilitats només definint el seu comportament en el mòdul *Raonador* i no haver de canviar res més.

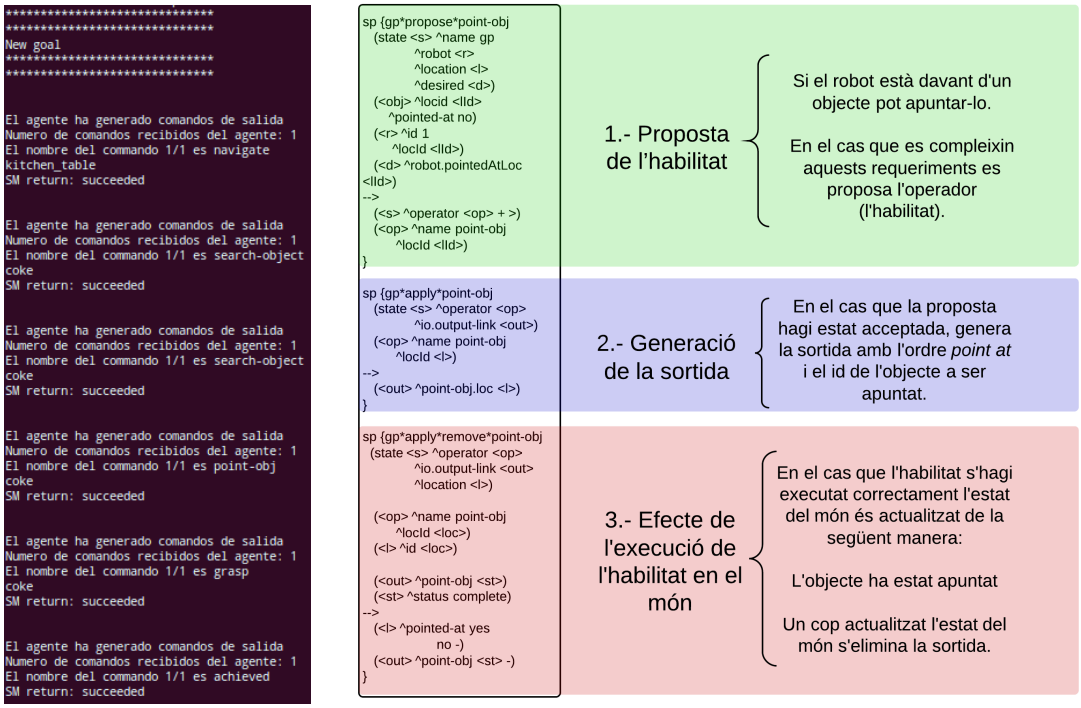
Figura 7.1: Seqüència d'accions realitzades pel robot REEM per l'ordre sense errors "*Bring me a drink*": Escoltar l'ordre, sol·licitar informació que falta i reconèixer resposta, go to kitchen, look for objects, detect energy drink, grasp coke, return to master, deliver energy drink.

Per tal d'estudiar aquesta característica s'ha realitzat el següent experiment: "Fer que el robot sigui capaç d'utilitzar una nova habilitat: point to the OBJECT", on OBJECT és qualsevol objecte. A continuació es descriu l'únic pas realitzat per introduir aquesta nova habilitat:

Tal com s'ha definit anteriorment, el submòdul *Soar* és l'encarregat de seleccionar les habilitats a ser executades en cada pas. Per tant, aquest mòdul ha de ser actualitzat amb la definició de la nova habilitat. Per fer-ho s'ha de definir:

1. Norma que proposa l'operador
2. Regla que envia la sortida que el robot executi aquesta habilitat si s'accepta el seu operador. Aquesta sortida conte el nom de l'habilitat juntament amb l'objecte ha ser apuntat.
3. Una o diverses regles que estableixen com afecta el món apuntar a un objecte.

Un cop completat aquest pas (veure Figura 7.2b) el robot ja és capaç d'utilitzar aquesta nova habilitat per completar goals. Per comprovar-ho, es va crear un món amb una coke a la cuina. En aquest món se li va demanar al robot "go to the kitchen, point the coke and grasp the coke"dos cops. El primer cop no es va introduir la nova habilitat i l'execució del goal va fallar en no poder arribar a l'estat desitjat del món. El segon cop sí que es va introduir la nova habilitat i el robot va completar el goal realitzant la següent seqüència d'accions: *go to the kitchen, search coke, point coke, grasp coke.* (veure Figura 7.2a)



(a) Sortida de l'ordre (b) Denició de l'habilitat *point at*

Figura 7.2: Sortida de l'ordre "go to the kitchen, point the coke and grasp the coke" & Denició de l'habilitat *point at*

En conclusió, aquest experiment demostra que aquesta arquitectura és prou canviable per a poder introduir noves habilitats sense haver de modificar cap aspecte anterior de l'arquitectura. Només s'ha de definir la nova habilitat en el submòdul *Soar*.

7.3 Incloure nous goals

Una de les grans característiques d'aquesta arquitectura és la facilitat de definir nous goals ha ser resolts pel robot sense haver de codificar nous plans ni noves habilitats. Per tal d'estudiar aquesta característica s'ha realitzat el següent experiment: "Fer que el robot sigui capaç de completar el nou goal: *empty the LOCATION*, on LOCATION és qualsevol espai de la casa". A continuació es descriuen tots els passos realitzats per introduir aquest nou goal:

1. En primer lloc, la gramàtica de l'extractor semàntic ha de ser actualitzada. Aquest pas és necessari per poder fer que el robot entengui la nova orde. Així doncs, ha d'introduir-se la paraula "empty"en la gramàtica i identificar-la com una acció a realitzar.

2. El segon pas consisteix a definir l'estat del món desitjat per aquest goal. Tal com s'ha definit anteriorment, el submòdul *Compilador* és l'encarregat de produir el goal en un format comprensible per Soar. Per tant, aquest mòdul ha de ser actualitzat amb la definició del nou estat desitjat. Per fer-ho s'ha d'afegir en el *Compilador* l'existència d'un nou goal anomenat "empty" i definir-lo com un estat del món on no es troba cap objecte en una localitat donada.

Un cop completats aquests dos passos el robot ja és capaç de resoldre aquest nou goal. Per comprovar-ho, es va crear un món amb una taula a la cuina amb una melmelada sobre d'ella. També es va definir una nova localització anomenada "trash bin"on el robot ha de desar tots els objectes retirats. En aquest món se li va demanar al robot "empty the kitchen table". La seqüència de les accions realitzades pel robot va ser la següent: *go to the kitchen table, search objects, grasp marmalade, go to the trash bin, deliver marmalade.* (Figura 7.3)

En resum, és clar ara, que aquesta arquitectura és prou flexible i canviable per compilar nous goals només realitzant dues actualitzacions simples sense haver de codificar nous plans ni noves habilitats.

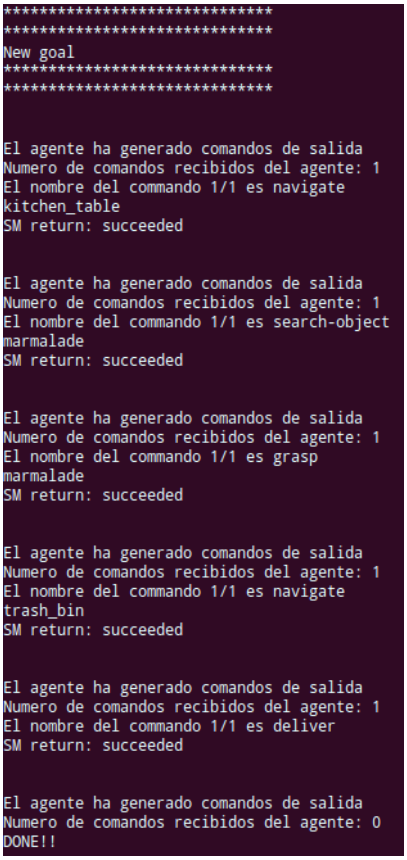


Figura 7.3: Sortida de l'ordre "empty the kitchen table"

7.4 Robot agnocstic

L'arquitectura implementada és *robot agnostic*, és a dir, que tota l'arquitectura pot ser instal·lada en qualsevol robot sense necessitat de modificar-la, no depèn del hardware. La demostració és teòrica. Les úniques parts de l'arquitectura que interactuen de forma directa amb el robot són el *ASR* i la *Interfície*. A més a més, ROS (el framework utilitzat) garanteix abstracció del hardware.

Així doncs, per garantir que es pugui instal·lar l'arquitectura en un robot donat, només és necessari que aquest garanteixi que el mòdul *ASR* tingui accés al micròfon i que les habilitats del robot proporcionin les entrades i sortides adequades en el format esperat per l'arquitectura.

Capítol 8

Conclusions i futures investigacions

En aquest treball final de grau s'ha implementat una arquitectura cognitiva basada en Soar per la resolució de tasques complexes. Suposant un robot de servei dotat de múltiples habilitats com desplaçar-se, reconèixer objectes, etc. El robot, utilitzant aquestes habilitats, gràcies a aquesta arquitectura, pot completar una tasca complexa donada en llenguatge natural que requereix aquestes habilitats per a ser resolta. Aquesta implementaicó suposa un nou enfocament per la resolució de tasques complexes en la robòtica de servei.

L'enfocament cognitiu de l'arquitectura permet evitar haver de fer planificació en el sentit clàssic. Aquest enfocament està basat en solucionar la situació donada en cada moment, no planifica tota l'execució de la tasca. L'arquitectura va resolent la solució actual pas a pas fins a aconseguir completar el goal (si és realitzable). Aquest comportament tan humà permet al robot poder-se adaptar a nous gols i situacions fàcilment perquè no necessita informació completa prèvia de l'entorn; cada cop que es troba amb un problema actua en conseqüència per solucionar-lo.

L'arquitectura original Soar no pot detectar si un goal és realitzable o no. Si no ho és, Soar continuarà intentant completar-lo i, en conseqüència, seguirà enviant infinites ordres d'execució d'habilitats al robot. En la implementació d'aquest treball s'ha restringit el conjunt de possibles goals (veure Apèndix B i C) que pot rebre el robot en el mòdul *Extractor Semàntic*. Així doncs, aquesta arquitectura garanteix que totes les ordres acceptades per l'arquitectura són realitzables.

Tota l'arquitectura és completament robot agnòstic i, per tant, pot ser adaptada a qualsevol altre robot (veure Secció 7.4). A més a més, afegir i eliminar habilitats és tan fàcil com definir l'estat final del món desitjat, tal com s'ha vist en la Secció 7.3. Aquestes característiques fan aquesta arquitectura extremadament adaptable i fàcil d'implementar en qualsevol robot. En el cas d'introduir-se en un

nou robot s'hauria d'actualitzar l'ontologia d'objectes i localitzacions, la gramàtica del *NLU* i el *Compilador* amb les noves possibles ordres, definir les habilitats del robot a *Soar* i finalment definir la crida de les habilitats en la *interfície*.

En els experiments realitzats en entorns reals s'ha detectat que els punts més crítics són la detecció automàtica de la parla i el reconeixement d'objectes. Aquest fet és dona perquè aquests dos sistemes depenen de l'escala del món coneguda pel robot. Com més gran és el coneixement del robot del món, major és la taxa d'error d'aquests sistemes. La detecció automàtica de la parla es veu afectada perquè el volum de vocabulari augmenta i en conseqüència la probabilitat de confusió del sistema augmenta, disminuint així la taxa de reconeixements correctes. El mateix efecte es produeix amb el reconeixement d'objectes, en augmentar el nombre de possibles objectes a reconèixer decrementa la taxa de reconeixement.

La implementació actual pot millorar-se en termes de robustesa resolent dos principals problemes. El primer ve donat quan una habilitat no es pot completar de forma satisfactòria temporalment (per exemple, el robot no és capaç d'arribar a una posició en l'espai perquè està ocupada). En aquest cas l'execució de l'habilitat retornarà que ha fallat i l'arquitectura implementada no té forma de descobrir el motiu de la fallida. Per tant, no és capaç de resoldre el problema de forma directa. Aquest desconeixement provoca que l'arquitectura detecti que l'estat del món no ha canviat i torni a intentar seleccionar la mateixa acció. Aquest comportament podria provocar un bucle infinit de reintents que s'ha evitat limitant els reintents a 3. El segon problema ve provocat per la impossibilitat de resoldre tasques on l'ordre conté errors (per exemple, "deliver the coke to yourself"). Les versions futures de l'arquitectura haurien d'incloure aquesta funció mitjançant la inclusió d'ontologies semàntiques i relacions com Wordnet [25] i VerbNet [28], fent l'extracció semàntica més robusta.

Finalment, també es podria extendre aquest treball explotant al màxim la funcionalitat *Reinforcement Learning* de Soar per a goals més complexos que involucrin un gran nombre de passos. Aquesta funcionalitat permetria minimitzar el nombre de passos proposats per assolir un goal, apropant-se així al camí òptim.

Capítol 9

Referències

[1] T. Kawahara A. Lee i K. Shikano. "Julius — an open source real-time large vocabulary recognition engine". A: *Proc. European Conference on Speech Communication and Technology (EUROSPEECH)*. 2001, pàg. 1691 - 1694.

[2] John R. Anderson. "ACT: A Simple Theory of Complex Cognition". A: *American Psychologist* (1995).

[3] Ruzena Bajcsy et al. "Landscan: A natural language and computer vision system for analyzing aerial images". A: *Proceedings of the 9th international joint conference on Artificial intelligence-Volume 2*. Morgan Kaufmann Publishers Inc. 1985, pàg. 919 - 921.

[4] Avron Barr. "Natural language understanding". A: *AI Magazine* 1.1 (1980), pàg. 5.

[5] Michael Beetz, M Lorenz i Moritz Tenorth. "CRAM — A C ognitive R obot A bstract M achine for Everyday Manipulation in Human Environments". A: *International Conference on Intelligent Robots and Systems (IROS)*. 2010.

[6] Michael Beetz et al. "Robotic Roommates Making Pancakes". A: *11th IEEE-RAS International Conference on Humanoid Robots*. Bled, Slovenia, 2011.

[7] Xiaoping Chen et al. "Developing High-level Cognitive Functions for Service Robots". A: *AAMAS '10 Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems* 1 (2010), pàg. 989 - 996.

[8] Scott D. Hanford. "A Cognitive Robotic System Based on Soar". Tesi doct. 2011.

[9] Stephen A. Zahorian Hongbing Hu. "Dimensionality Reduction Methods for HMM Phonetic Recognition". A: *ICASSP 2010* (2010).

[10] Thomas M Howard, Stefanie Tellex i Nicholas Roy. "A Natural Language Planner Interface for Mobile Manipulators". A: ().

[11] Andrew Howes i Richard M. Young. "The Role of Cognitive Architecture in Modeling the User: Soar's Learning Mechanism". A: *Human-Computer Interaction* 12.4 (1997), pàg. 311 - 343. URL: http://dblp.uni-trier.de/db/journals/hhci/hhci12.html\#HowesY97.

[12] John HL: Hansen John R. Deller John G. Proakis. "Discrete-time processing of speech signals". A: *IEEE, 2000* (2000).

[13] Randolph M Jones. "An Introduction to Cognitive Architectures for Modeling and Simulation". A: 1987 (2004).

[14] B. H. Juang i L. R. Rabiner. "Hidden Markov Models for Speech Recognition". A: *Technometrics* 33.3 (1991), pàg. 251 - 272. DOI: 10.1080/00401706.1991.10484833. eprint: http://www.tandfonline.com/doi/pdf/10.1080/00401706.1991.10484833. URL: http://www.tandfonline.com/doi/abs/10.1080/00401706.1991.10484833.

[15] Troy Dale Kelley. "Developing a Psychologically Inspired Cognitive Architecture for Robotic Control : The Symbolic and Subsymbolic Robotic Intelligence Control System". A: *Internation Journal of Advanced Robotic Systems* 3.3 (2006), pàg. 219 - 222.

[16] Thomas Kollar et al. "Generalized Grounding Graphs: A Probabilistic Framework for Understanding Grounded Language". A: ().

[17] Thomas Kollar et al. "Toward understanding natural language directions". A: *Proceeding of the 5th ACM/IEEE international conference on Human-robot interaction - HRI '10* (2010), pàg. 259. DOI: 10.1145/1734454.1734553. URL: http://portal.acm.org/citation.cfm?doid=1734454.1734553.

[18] John E. Laird, Allen Newell i Paul S. Rosenbloom. "Soar: An Architecture for General Intelligence". A: *Artificial Intelligence* 33.1 (set. de 1987), pàg. 1 - 64. ISSN: 0004-3702. DOI: 10.1016/0004-3702(87)90050-6. URL: http://dx.doi.org/10.1016/0004-3702(87)90050-6.

[19] John E Laird i Robert E Wray III. "Cognitive Architecture Requirements for Achieving AGI". A: *Proceedings of the Third Conference on Artificial General Intelligence*. 2010.

[20] John E Laird et al. "Cognitive Robotics using the Soar Cognitive Architecture". A: *Proc. of the 6th Int. Conf.on Cognitive Modelling*. 2004, pàg. 226 - 230.

[21] Pat Langley, John E. Laird i Seth Rogers. "Cognitive architectures: Research issues and challenges". A: *Cognitive Systems Research* 10.2 (jun. de 2009), pàg. 141 - 160. ISSN: 13890417. DOI: 10.1016/j.cogsys.2006.07.004. URL: http://linkinghub.elsevier.com/retrieve/pii/S1389041708000557.

[22] Stephen E Levinson, Lawrence R Rabiner i Man Mohan Sondhi. "An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition". A: *Bell System Technical Journal* 62.4 (1983), pàg. 1035 - 1074.

[23] Austin Marshall. "Artificial Neural Network for Speech Recognition". Tesi doct. 2005.

[24] Robert Cecil Martin. *Agile software development: principles, patterns, and practices*. Prentice Hall PTR, 2003.

[25] George A. Miller. "WordNet: A Lexical Database for English". A: *Communications of the ACM* 38.11 (1995), pàg. 39 - 41.

[26] Shiwali Mohan i John E Laird. "Situated Comprehension of Imperative Sentences in Embodied , Cognitive Agents". A: (2012).

[27] Shelley Nason i John E. Laird. "Soar-RL: Integrating Reinforcement Learning with Soar". A: *Cognitive Systems Research* 6.1 (2005), pàg. 51 - 59. URL: `http://dblp.uni-trier.de/db/journals/cogsr/cogsr6.html#NasonL05`.

[28] Martha Palmer et al. "Extensive Classifications of English verbs". A: *Proceedings of the 12th EURALEX International Congress*. 2006.

[29] Bo Pang, Lillian Lee i Shivakumar Vaithyanathan. "Thumbs up?: sentiment classification using machine learning techniques". A: *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. Association for Computational Linguistics. 2002, pàg. 79 - 86.

[30] HUANG Pei-hong. "Formalization of Natural Language Understanding". A: *Computer Engineering & Science* 6 (2007), pàg. 033.

[31] Lawrence R Rabiner i Biing-Hwang Juang. *Fundamentals of speech recognition*. Vol. 14. PTR Prentice Hall Englewood Cliffs, 1993.

[32] Mosur K. Ravishankar. *Efficient algorithms for speech recognition*. Inf. tèc. 1996.

[33] Wolfgang Schleidt i Thomas R. Shultz. "Newell: Unified Theories of Cognition". A: *Behavioral and Brain Sciences* (1992), pàg. 456 - 457.

[34] SoarTechnology. *Soar: A Functional Approach to General Intelligence*. Inf. tèc. 2002, pàg. 2 - 4.

[35] Terrence C Stewart i Robert L West. "Deconstructing ACT-R". A: *Proceedings of the Seventh International Conference on Cognitive Modeling*. 2006.

[36] Peter Norvig Stuart Russell. *Artificial Intelligence: A Modern Approach*.

[37] Wolfgang Wahlster et al. "OVER-ANSWERING YES-NO QUESTIONS". A: *IJCAI-83: proceedings of the Eighth International Joint Conference on Artificial Intelligence, 8-12 August 1983, Karlsruhe, West Germany*. Vol. 2. Joint Conferences. 1983, pàg. 643.

Apèndix A

Glossari d'abreviatures

ACT-R: Adaptive Control of Thought—Rational
API: Application Programming Interface
ASR: Automatic Speech Recognition
BSD: Berkeley Software Distribution
CMU: Carnegie Mellon University
CRAM: Cognitive Robotic Abstract Machine
HMM: Hidden Markov Model
HRI: Human–robot interaction
IBM: International Business Machines
IFR: International Federation of Robotics
IRI: Institut de Robòtica i Informàtica Industrial
LCM: Lightweight Communications and Marshalling
MIT: Massachusetts Institute of Technology
NLU: Natural Language Understanding
ROS: Robot Operating System
SS-RICS: Symbolic and Subsymbolic Robotic Intelligence Control System
UPC: Universitat Politècnica de Catalunya

Apèndix B

Llista de possibles goals de *Categoria I*

En aquest apèndix es llisten tots els possibles goals de *Categoria I* que l'arquitectura implementada pot reconèixer i resoldre.

Move to the LOCATION, then go to the LOCATION and exit the apartment. Move to the LOCATION, then move to the LOCATION and leave the apartment. Move to the LOCATION, then move to the LOCATION and exit the apartment. Move to the LOCATION, then navigate to the LOCATION and leave the apartment. Move to the LOCATION, then navigate to the LOCATION and exit the apartment. Navigate to the LOCATION, then go to the LOCATION and leave the apartment. Navigate to the LOCATION, then go to the LOCATION and exit the apartment. Navigate to the LOCATION, then move to the LOCATION and leave the apartment. Navigate to the LOCATION, then move to the LOCATION and exit the apartment. Navigate to the LOCATION, then navigate to the LOCATION and leave the apartment. Navigate to the LOCATION, then navigate to the LOCATION and exit the apartment. Go to the LOCATION, get the ITEM, and leave the apartment. Go to the LOCATION, get the ITEM, and exit the apartment. Go to the LOCATION, take the ITEM, and leave the apartment. Go to the LOCATION, take the ITEM, and exit the apartment. Go to the LOCATION, grasp the ITEM, and leave the apartment. Go to the LOCATION, grasp the ITEM, and exit the apartment. Move to the LOCATION, get the ITEM, and leave the apartment. Move to the LOCATION, get the ITEM, and exit the apartment. Move to the LOCATION, take the ITEM, and leave the apartment. Move to the LOCATION, take the ITEM, and exit the apartment. Move to the LOCATION, grasp the ITEM, and leave the apartment. Move to the LOCATION, grasp the ITEM, and exit the apartment. Navigate to the LOCATION, get the ITEM, and leave the apartment. Navigate to the LOCATION, get the ITEM, and exit the apartment. Navigate to the LOCATION, take the ITEM, and leave the apartment.

Navigate to the LOCATION, take the ITEM, and exit the apartment. Navigate to the LOCATION, grasp the ITEM, and leave the apartment. Navigate to the LOCATION, grasp the ITEM, and exit the apartment. Go to the LOCATION, introduce yourself, and leave the apartment. Go to the LOCATION, introduce yourself, and exit the apartment. Go to the LOCATION, tell something about yourself, and leave the apartment. Go to the LOCATION, tell something about yourself, and exit the apartment. Move to the LOCATION, introduce yourself, and leave the apartment. Move to the LOCATION, introduce yourself, and exit the apartment. Move to the LOCATION, tell something about yourself, and leave the apartment. Move to the LOCATION, tell something about yourself, and exit the apartment. Navigate to the LOCATION, introduce yourself, and leave the apartment. Navigate to the LOCATION, introduce yourself, and exit the apartment. Navigate to the LOCATION, tell something about yourself, and leave the apartment. Navigate to the LOCATION, tell something about yourself, and exit the apartment. Go to the LOCATION, memorize the person, and recognize him. Go to the LOCATION, ask the person's name, and recognize him. Move to the LOCATION, memorize the person, and recognize him. Move to the LOCATION, ask the person's name, and recognize him. Navigate to the LOCATION, memorize the person, and recognize him. Navigate to the LOCATION, ask the person's name, and recognize him. Go to the LOCATION, introduce yourself, and follow the person in front of you. Go to the LOCATION, tell something about yourself, and follow the person in front of you. Move to the LOCATION, introduce yourself, and follow the person in front of you. Move to the LOCATION, tell something about yourself, and follow the person in front of you. Navigate to the LOCATION, introduce yourself, and follow the person in front of you. Navigate to the LOCATION, tell something about yourself, and follow the person in front of you. Memorize the person, follow the person in front of you, and leave the apartment. Memorize the person, follow the person in front of you, and exit the apartment. Ask the person's name, follow the person in front of you, and leave the apartment. Ask the person's name, follow the person in front of you, and exit the apartment. Find a person, retrieve the ITEM from the LOCATION, and leave the apartment. Find a person, retrieve the ITEM from the LOCATION, and exit the apartment. Find a person, bring the ITEM from the LOCATION, and leave the apartment. Find a person, bring the ITEM from the LOCATION, and exit the apartment. Go to the LOCATION, find a person, and retrieve the ITEM from the LOCATION. Go to the LOCATION, find a person, and bring the ITEM from the LOCATION. Move to the LOCATION, find a person, and retrieve the ITEM from the LOCATION. Move to the LOCATION, find a person, and bring the ITEM from the LOCATION. Navigate to the LOCATION, find a person, and retrieve the ITEM from the LOCATION. Navigate to the LOCATION, find a person, and bring the ITEM from the LOCATION. Memorize the person, follow the person in front of you, and leave the apartment. Memorize the person, follow the person in front of you, and exit the apartment. Ask the person's name, follow the person in

front of you, and leave the apartment. Ask the person's name, follow the person in front of you, and exit the apartment. Go to the LOCATION, find a person, and introduce yourself. Go to the LOCATION, find a person, and tell something about yourself. Move to the LOCATION, find a person, and introduce yourself. Move to the LOCATION, find a person, and tell something about yourself. Navigate to the LOCATION, find a person, and introduce yourself. Navigate to the LOCATION, find a person, and tell something about yourself. Get the ITEM, bring it to the LOCATION, and introduce yourself. Get the ITEM, bring it to the LOCATION, and tell something about yourself. Get the ITEM, carry it to the LOCATION, and introduce yourself. Get the ITEM, carry it to the LOCATION, and tell something about yourself. Take the ITEM, bring it to the LOCATION, and introduce yourself. Take the ITEM, bring it to the LOCATION, and tell something about yourself. Take the ITEM, carry it to the LOCATION, and introduce yourself. Take the ITEM, carry it to the LOCATION, and tell something about yourself. Grasp the ITEM, bring it to the LOCATION, and introduce yourself. Grasp the ITEM, bring it to the LOCATION, and tell something about yourself. Grasp the ITEM, carry it to the LOCATION, and introduce yourself. Grasp the ITEM, carry it to the LOCATION, and tell something about yourself. Go to the LOCATION, get the ITEM, and bring it to the LOCATION. Go to the LOCATION, get the ITEM, and carry it to the LOCATION. Go to the LOCATION, take the ITEM, and bring it to the LOCATION. Go to the LOCATION, take the ITEM, and carry it to the LOCATION. Go to the LOCATION, grasp the ITEM, and bring it to the LOCATION. Go to the LOCATION, grasp the ITEM, and carry it to the LOCATION. Move to the LOCATION, get the ITEM, and bring it to the LOCATION. Move to the LOCATION, get the ITEM, and carry it to the LOCATION. Move to the LOCATION, take the ITEM, and bring it to the LOCATION. Move to the LOCATION, take the ITEM, and carry it to the LOCATION. Move to the LOCATION, grasp the ITEM, and bring it to the LOCATION. Move to the LOCATION, grasp the ITEM, and carry it to the LOCATION. Navigate to the LOCATION, get the ITEM, and bring it to the LOCATION. Navigate to the LOCATION, get the ITEM, and carry it to the LOCATION. Navigate to the LOCATION, take the ITEM, and bring it to the LOCATION. Navigate to the LOCATION, take the ITEM, and carry it to the LOCATION. Navigate to the LOCATION, grasp the ITEM, and bring it to the LOCATION. Navigate to the LOCATION, grasp the ITEM, and carry it to the LOCATION. Go to the LOCATION, detect the ITEM, and get it. Go to the LOCATION, detect the ITEM, and take it. Go to the LOCATION, detect the ITEM, and grasp it. Go to the LOCATION, find the ITEM, and get it. Go to the LOCATION, find the ITEM, and take it. Go to the LOCATION, find the ITEM, and grasp it. Go to the LOCATION, identify the ITEM, and get it. Go to the LOCATION, identify the ITEM, and take it. Go to the LOCATION, identify the ITEM, and grasp it. Move to the LOCATION, detect the ITEM, and get it. Move to the LOCATION, detect the ITEM, and take it. Move to the LOCATION, detect the ITEM, and grasp it. Move to the

LOCATION, find the ITEM, and get it. Move to the LOCATION, find the ITEM, and take it. Move to the LOCATION, find the ITEM, and grasp it. Move to the LOCATION, identify the ITEM, and get it. Move to the LOCATION, identify the ITEM, and take it. Move to the LOCATION, identify the ITEM, and grasp it. Navigate to the LOCATION, detect the ITEM, and get it. Navigate to the LOCATION, detect the ITEM, and take it. Navigate to the LOCATION, detect the ITEM, and grasp it. Navigate to the LOCATION, find the ITEM, and get it. Navigate to the LOCATION, find the ITEM, and take it. Navigate to the LOCATION, find the ITEM, and grasp it. Navigate to the LOCATION, identify the ITEM, and get it. Navigate to the LOCATION, identify the ITEM, and take it. Navigate to the LOCATION, identify the ITEM, and grasp it. Go to the LOCATION, then go to the LOCATION and leave the apartment. Go to the LOCATION, then go to the LOCATION and exit the apartment. Go to the LOCATION, then move to the LOCATION and leave the apartment. Go to the LOCATION, then move to the LOCATION and exit the apartment. Go to the LOCATION, then navigate to the LOCATION and leave the apartment. Go to the LOCATION, then navigate to the LOCATION and exit the apartment. Move to the LOCATION, then go to the LOCATION and leave the apartment.

Apèndix C

Llista de possibles goals de *Categoria II*

En aquest apèndix es llisten tots els possibles goals de *Categoria II* que l'arquitectura implementada pot reconèixer i resoldre.

Bring a drink to a table. Bring a drink to a shelf. Bring a drink to a door. Bring a drink to an appliance. Bring a snack to a seat. Bring a snack to a table. Bring a snack to a shelf. Bring a snack to a door. Bring a snack to an appliance. Bring the kitchenary to a seat. Bring the kitchenary to a table. Bring the kitchenary to a shelf. Bring the kitchenary to a door. Bring the kitchenary to an appliance. Bring some food to a seat. Bring some food to a table. Bring some food to a shelf. Bring some food to a door. Bring some food to an appliance. Carry a drink to a seat. Carry a drink to a table. Carry a drink to a shelf. Carry a drink to a door. Carry a drink to an appliance. Carry a snack to a seat. Carry a snack to a table. Carry a snack to a shelf. Carry a snack to a door. Carry a snack to an appliance. Carry the kitchenary to a seat. Carry the kitchenary to a table. Carry the kitchenary to a shelf. Carry the kitchenary to a door. Carry the kitchenary to an appliance. Carry some food to a seat. Carry some food to a table. Carry some food to a shelf. Carry some food to a door. Carry some food to an appliance. Get me a drink. Get me a snack. Get me the kitchenary. Get me some food. Give me a drink. Give me a snack. Give me the kitchenary. Give me some food. Point at a seat. Point at a table. Point at a shelf. Point at a door. Point at an appliance. Go to an exit. Move to an exit. Navigate to an exit. Detect a drink. Detect a snack. Detect the kitchenary. Detect some food. Find a drink. Find a snack. Find the kitchenary. Find some food. Bring a drink to a seat.